Class 12

Answer Key

Computer Science

1. Boolean Algebra

Unsolved Questions

A. 1. b 2. b 3. c 4. a 5. d
6. b 7. d 8. d 9. c 10. b

B. 1. Absorption 2. A 3. X'' 4. A.B.C 5. Even
6. $(A' \vee B) \wedge (B' \vee A)$ 7. All 8. Converse

C. 1. 1. a. **Simple proposition:** It contains a single atomic statement. Example: "Planet Mars is named after the Roman God of war."

Compound proposition: It contains two or more simple propositions joined by special symbols called connectives. Example: "Mercury is the smallest planet in the solar system and it has no moon."

b. **Conjunction:** It is a binary connective as it joins two or more simple propositions. It results in true if all proposition variables are true. If any variable is false, the resulting output is false. It is represented by a dot (.) or $\wedge$.

Disjunction: It is also a binary connective that results in true if any one proposition is true. The output is false only when both propositions are false. It is represented by + or $\vee$.

c. **Conditional:** It is represented by the symbol $\rightarrow$. Implication results in false if the first proposition (if) is true but the second proposition (then) is false. It returns true in all other cases.

Biconditional: It represents the 'if and only if' condition. It returns true when both propositions have the same value and false if they have different values. It is represented by the $\leftrightarrow$ symbol.

d. **Maxterm:** It can be defined as the sum terms of all the variables present in the expression both in complemented and normal forms.

Minterm: It can be defined as the product terms of all the variables present in the expression both in complemented and normal forms.

e. **SOP:** In SOP, the Boolean expression is formed by ORing multiple ANDed terms.

POS: In POS, the Boolean expression is formed by ANDing multiple ORed terms.

2. a. $Q \rightarrow (P \wedge Q)$ b. $P \wedge Q$ c. $Q \leftrightarrow P$ d. $P' \vee Q'$ e. $Q \rightarrow R$

3. a. It is not raining heavily or there is no prediction for cyclone

b. It is not raining heavily or there is no prediction for cyclone and there will be no damage of life and property

c. If and only if it is not raining heavily then there is no prediction for cyclone

d. It is false that it is raining heavily and there is a prediction for cyclone and there will be damage to life property

e. If there is no prediction for cyclone then there will be no damage to life and property

4. i. **Converse:** You will crack IIT if you work hard
Inverse: If you do not work hard then you will not crack IIT
Contrapositive: You will not crack IIT if you do not work hard

ii. **Converse:** If it is rose then the flower has a beautiful smell
Inverse: If the flower does not have a beautiful smell then it is not rose
Contrapositive: If it is not rose then the flower does not have a beautiful smell

iii. **Converse:** If I will wear my red gown then I am invited to a party
Inverse: If I am not invited to a party then I will not wear my red gown
Contrapositive: If I will not wear my red gown then I am not invited to a party

iv. **Converse:** If we will celebrate then India will win the world cup cricket match
Inverse: If India does not win the world cup cricket match then we will not celebrate
Contrapositive: If we will not celebrate then India will not win the world cup cricket match

v. **Converse:** If it is a palindrome then the number is equal to its reverse
Inverse: If the number is not equal to its reverse then it is not a palindrome
Contrapositive: If it is not a palindrome then the number is not equal to its reverse

5. a.

a	b	$a \leftrightarrow b$	$a \wedge b$	$(a \leftrightarrow b) \rightarrow (a \wedge b)$
0	0	1	0	0
0	1	0	0	1
1	0	0	0	1
1	1	1	1	1
It is a contingency				

b.

a	b	$\sim a \rightarrow b$	$b \rightarrow (\sim a \rightarrow b)$	$b \rightarrow (\sim a \rightarrow b) \leftrightarrow a$
0	0	0	1	0
0	1	1	1	0
1	0	1	1	1
1	1	1	1	1
It is a contingency				

c.

a	b	$a \wedge b$	$\sim a \wedge \sim b$	$(a \wedge b) \rightarrow (\sim a \wedge \sim b)$
0	0	0	1	1
0	1	0	0	1
1	0	0	0	1
1	1	1	0	0
It is a contingency				

d.

a	b	c	$a \rightarrow b$	$b \rightarrow c$	$[(a \rightarrow b) \rightarrow (\sim b \rightarrow c)]$	$[(a \rightarrow b) \rightarrow (\tilde{}b \rightarrow c)] \rightarrow a$
0	0	0	1	1	1	0
0	0	1	1	1	1	0
0	1	0	1	0	0	1
0	1	1	1	1	1	0
1	0	0	0	1	1	1
1	0	1	0	1	1	1
1	1	0	1	0	0	1
1	1	1	1	1	1	1
It is a contingency						

e.

a	b	$a \wedge 0$	$\sim a \wedge b$	$(a \wedge 0) \rightarrow (\sim a \wedge b)$
0	0	0	0	1
0	1	0	1	1
1	0	0	0	1
1	1	0	0	1
It is a tautology				

f.

a	b	$a \rightarrow b$	$a \wedge (a \rightarrow b)$	$a \wedge (a \rightarrow b) \rightarrow b$
0	0	1	0	1
0	1	1	0	1
1	0	0	0	1
1	1	1	1	1
It is a tautology				

g.

a	b	$a \leftrightarrow b$	$\sim a \leftrightarrow \sim b$	$(a \leftrightarrow b) \wedge (\sim a \leftrightarrow \sim b)$
0	0	1	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	1	1
It is a contingency				

h.

a	b	$a \wedge b$	$\sim a \wedge b$	$a \wedge \sim b$	$(a \wedge b) \rightarrow (\sim a \wedge b)$	$(a \wedge b) \rightarrow (\sim a \wedge b) \rightarrow (a \wedge \sim b)$
0	0	0	0	0	1	0
0	1	0	1	0	1	0
1	0	0	0	1	1	1
1	1	1	0	0	0	1
It is a contingency						

i.

a	b	~(a ∨ b)	(a ∨ b)	~(a ∨ b) ∧ (a ∨ b)
0	0	1	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	1	0
It is a contradiction				

j.

a	b	c	~(a ∧ b ∧ c)	a ∧ b ∧ c	~(a ∧ b ∧ c) ∨ (a ∧ b ∧ c)
0	0	0	1	0	1
0	0	1	1	0	1
0	1	0	1	0	1
0	1	1	1	0	1
1	0	0	1	0	1
1	0	1	1	0	1
1	1	0	1	0	1
1	1	1	0	1	1
It is a tautology					

6.

P	Q	P → Q	(P → Q) → P'	~ ((P → Q) → P')
0	0	1	1	0
0	1	1	1	0
1	0	0	1	0
1	1	1	0	1

7. a. X.Y + X'.Z + Y.Z

= X.Y + X'.Z + Y.Z.1

= X.Y + X'.Z + Y.Z.(X+X')

= X.Y + X'.Z + X.Y.Z + X'.Y.Z

= X.Y.(1+Z) + X'.Z.(I+Y)

= X.Y + X'.Z

b. A.(B+C.(A+B.C)')'

= A. (B + C.(A' . (B.C)')'

= A.(B + C.(A'.(B'+C'))'

= A.(B + A'.B'.C + A'.C'.C)'

= A.(B + A'.B'.C+0)'

= A.(B + A'.B'.C)'

= A.(B'.(A''+B''+C)

= A. B'.(A+B+C')

=A.B'.A + A.B'.B + A.B'.C'

=A.B' + 0 + A.B'.C'

= A.B' + A.B'.C'

= A.B'(1+C')

= A.B'

c. (X'+Y'+Z).(X'+Y'+Z') + (X+Y+Z).(X+Y+Z')

= (X'+Y') + Z.Z' + (X+Y) + Z.Z'

= (X'+Y') + 0 + (X+ Y)+0

= X' + Y' + X + Y

= (X'+X) + (Y'+Y)

=1

d. (X + X'.Y).(X'+Y')

= (X+X').(X+Y).(X'+Y')

= 1.(X+Y).(X'+Y')

= X.X'+X.Y'+X'.Y+Y.Y'

=0+X'.Y + X.Y'+0

= X'.Y + X.Y'

e. A.B.C + A.B.C' + A'.B.C + A.B'.C

= A.B.C + A.B.C + A.B.C + A.B.C' + A'.B.C + A.B'.C

= (A.B.C'+A.B.C) + (A'.B.C+A.B.C) + (A.B'.C + A.B.C)

= A.B.(C'+C) + B.C.(A'+A) + A.C.(B'+B)

=A.B.1 + B.C.1 + A.C.1

= A.B + B.C + A.C

f. X'.Y + X'.Y' + (Z.Z)' + Z

= X' (Y+Y') + Z' + Z

= X'.1 + 1

= 1

g. A.B + A.B' + A'.B'

= A.B + A.B' + A.B' + A'.B'

= A.(B+B') + B'.(A+A')

= A.1 + B'.1

= A+ B'

h. (X' + (X.Y)' + Z)'

= (X' + X' + Y' + Z)'

= (X' + Y' + Z)'

= X".Y".Z'

=X.Y.Z'

i. A.(A' + B).C.(A + B)

= A.C.(B + A.A')

= A.C.(B + 0)

=A.B.C

j. (x' + z) + [(y' + z).(x' + y)]'

= x' + z + [(x'.y' + y.y' + x'.z + y.z)]'

= x' +z+ (x'.y' + 0 + x'.z + y.z)'

= x' + z +[(x'.y')'.(x'.z)'.(y.z)']

= x' + z + [(x" + y").(x" + z').(y'+z')]

= x' + z + [(x+y).(x + z').(y' + z')]

= x' + z + [(x + y.z').(y'+z')]

= x' + z + (x'.y' + y.y'.z' + x.z' + y.z'.z')

= x' + z + (x'.y' + 0 + x.z' + y.z')

= x' + x'.y' + z + x.z' + y.z'

= x'(1 + y') + z + x.z' + z + y.z'

= x' + (z + z').(x + z) + (z+z').(z + y)

= x' + 1.(x + z) + 1.(z + y)

= x' + x + z + z + y

= 1 + z + y

= 1

8. a. F(A,B,C,D) = Σ(2, 3, 4, 5, 6, 7, 8, 10, 11)

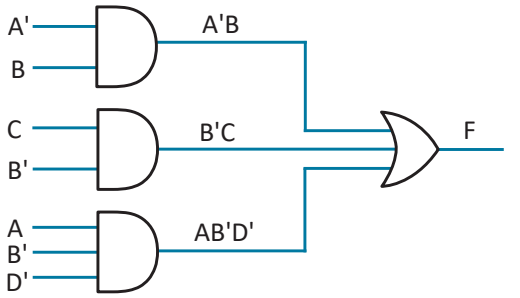
	CD'	CD	CD	CD'
A'B'	0 0	1 0	3 1	2 1
A'B	4 1	5 1	7 1	6 1
AB	12 0	13 0	15 0	14 0
AB'	8 1	9 0	11 1	10 1

There are two quads and one pair.

Quad 1(m2 + m3 + m10 m11) = B'C Quad2(m4+m5+m6+m7) = A'B

Pair(m8+m10) = AB'D'

Hence F(A, B, C, D) = B'C + A'B + AB'D'



b. F(P,Q,R,S) = π(0, 1, 2, 4, 5, 6, 8, 10)

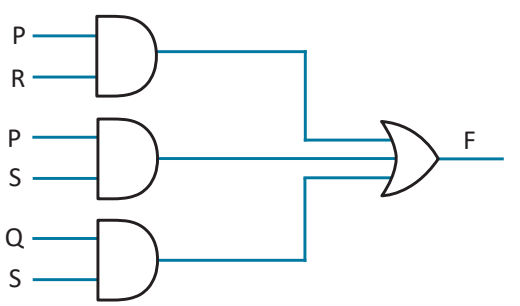
	R+S	R+S'	R'+S'	R'+S
P+Q	0 0	1 0	3 1	2 0
P+Q'	4 0	5 0	7 1	6 0
P'+Q'	12 1	13 1	15 1	14 1
P'+Q	8 0	9 1	11 1	10 0

There are three quads:

Quad 1: (M0 M1 M4 M5) = P + R Quad 2 : (M0 M2 M4 M6) = P + S

Quad 3: (M0 M2 M8 M10) = Q + S

Hence F(P, Q, R, S) = (P + R).(P + S).(Q + S)



9. a.

	C'.D'	C'.D	C.D	C.D'
A'.B'	0 0	1 1	1 3	0 2
A'.B	0 4	1 5	1 7	0 6
A.B	0 12	0 13	1 15	1 14
A.B'	1 8	1 9	1 11	1 10

Quad 1: m_1m_3,m_5,m_7 = A'.D
Quad 2: m_8,m_9,m_{11},m_{12} = A.B'
Quad 3: $m_{11},m_{12},m_{14},m_{15}$ = A.C
Reduced SOP expression: A'.D + A.B'+ A.C

	R+S	R+S'	R'+S'	R'+S
P+Q	1 0	1 1	1 3	1 2
P+Q'	0 4	1 5	0 7	0 6
P'+Q'	0 12	1 13	0 15	0 14
P'+Q	1 8	1 9	0 11	0 10

Quad 1: $m_4+m_6+m_{12}+m_{14}$ = Q'+S
Quad 2: $m_{10}+m_{11}+m_{14}+m_{15}$ = P'+R'
Quad 3: $m_6+m_7+m_{14}+m_{15}$ = Q'+R'
Reduced SOP expression:
(Q'+S).(P'+R').(Q'+R')

2. Computer Hardware

Unsolved Questions

A. 1. c 2. d 3. b 4. a 5. b
6. d 7. b 8. d 9. c 10. d

B. 1. A'.B'+A.B 2. OR 3. X'.Y+X.Y' 4. fundamental 5. odd
6. A'.B'+A.B 7. half adder 8. all 9. converse 10. A.B

C. 1. (i)

NAND	NOR
a. Complement of AND gate	a. Complement of OR gate
b. Output is 0 when all inputs are 1	b. Output is 1 when all inputs are 0

(ii)

Encoder	Decoder
a. Converts Decimal, Octal, Hexadecimal to Binary	a. Converts Binary to Decimal, Octal or Hexadecimal
b. OR gate used in circuit	b. AND gate used in circuit

(iii)

Decoder	Multiplexer
1. Decoder converts binary data to Octal, Decimal and Hexadecimal form	1. Multiplexer selects a single output from multiple binary inputs
2. Decoder is used to interpret coded data	2. MUX is used to transmit data

(iv)

AND gate	OR gate
a. Produces true when all conditions are true	a. Produces true when any one condition is true
b. Boolean expression is A.B	b. Boolean expression is A+B

(v)

Half Adder	Full Adder
The half adder is a combinational circuit that is designed to perform the addition of two bits and produce the two binary outputs as sum (S) and carry (C) bits.	The full adder is a combinational circuit that is designed to perform the addition of three bits and produce the two binary outputs as sum (S) and carry (C) bits.

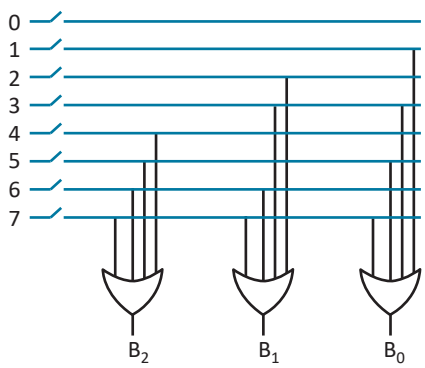
2. (i) A.B (ii) (A.B.C)' (iii) A.B + C' (iv) ((A.B.C)' . (A.B + C'))'

3. (i) A.B.C (ii) (A.C + B)' (iii) (A.B.C + (A.C + B')')'

4. (i) The truth table is given below:

Octal Number	B_2	B_1	B_0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

The logic circuit diagram of an Octal to Binary Encoder is as follows:

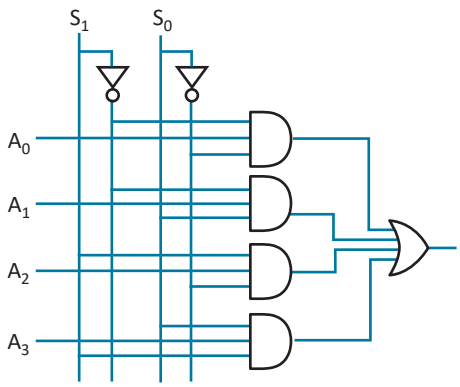


(ii) Multiplexer is a combinational circuit which selects a single output from a set of inputs. A multiplexer has 2^n input lines and one output line, where n is the number of selection lines.

Uses of Multiplexer

1. **Communication System:** Multiplexer allows transmission of different types of data such as audio and video at the same time using a single transmission line.
2. **Telephone Network:** In a telephone network, a Multiplexer help to isolate multiple audio signals sent through a single transmission line, to reach the intended recipients.
3. **Remote:** Remote control used in TV, Cars, and A.C select a specific signal from many available options.

Logic circuit diagram of a 4x1 MUX is



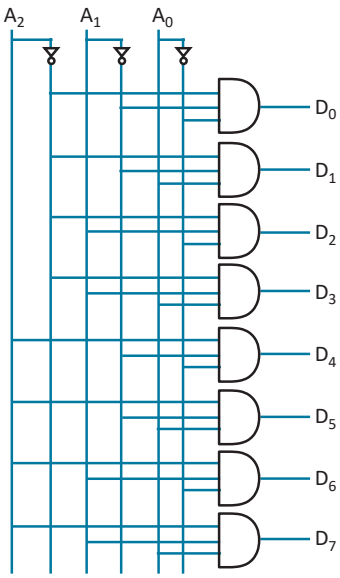
5.

Decoder	Multiplexer
1. Decoder converts binary data to Octal, Decimal and Hexadecimal form	1. Multiplexer selects a single output from multiple binary inputs
2. Decoder is used to interpret coded data	2. MUX is used to transmit data

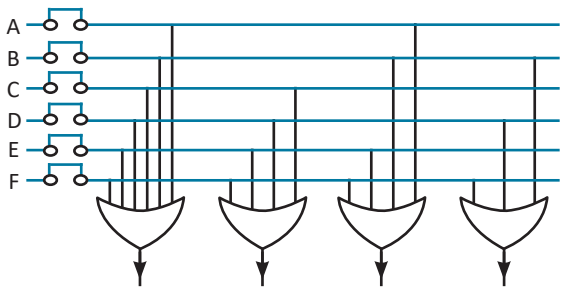
Truth table of a 3 to 8 decoder circuit

A_2	A_1	A_0	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Logic circuit of 3 to 8 decoder circuit



6. (i) An Encoder is a combinational circuit which inputs 2n or fewer lines and outputs n lines.
Circuit diagram of a A – F Encoder:



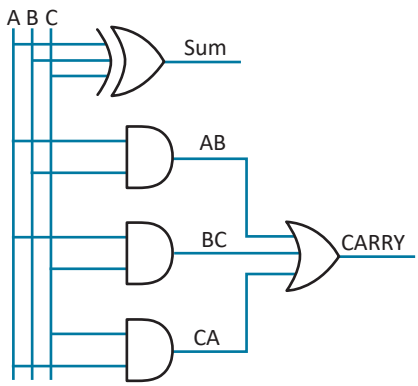
Application of a Multiplexer:

1. Used for Routing of signals
2. Data Transmission
3. Telephone Exchanges / TV etc

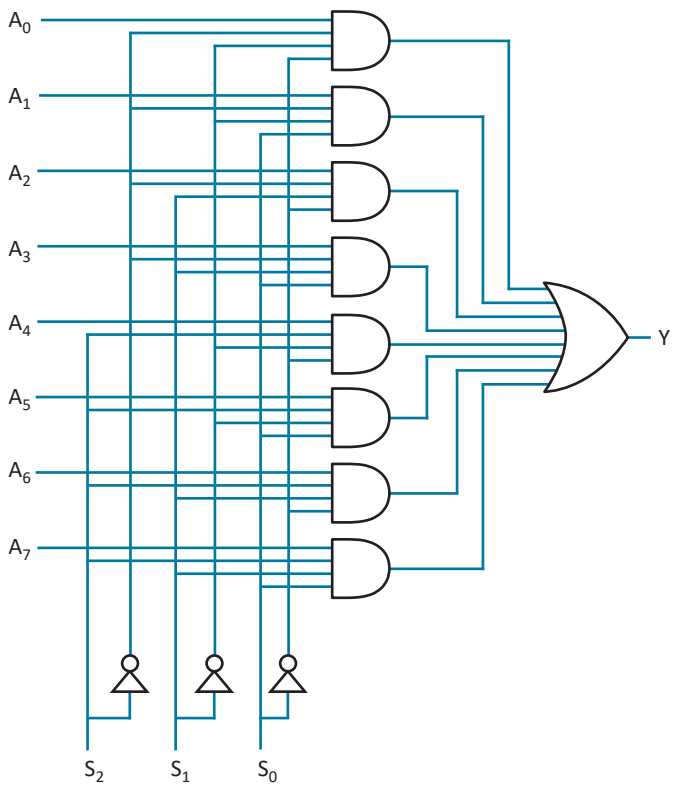
(ii) **Half Adders:** It is a combinational circuit which adds two input binary bits and outputs two binary bits.
Full Adders: It is a combinational circuit which adds three input binary bits and outputs two binary bits.

Circuit of a Full Adder: SUM=A⊕B⊕C

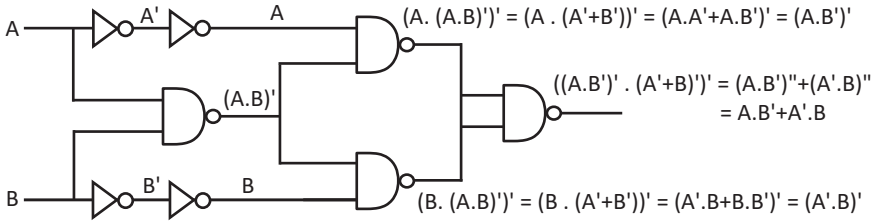
CARRY = AB + BC + AC



7. The logic circuit diagram of a 8 x 1 MUX is given below:

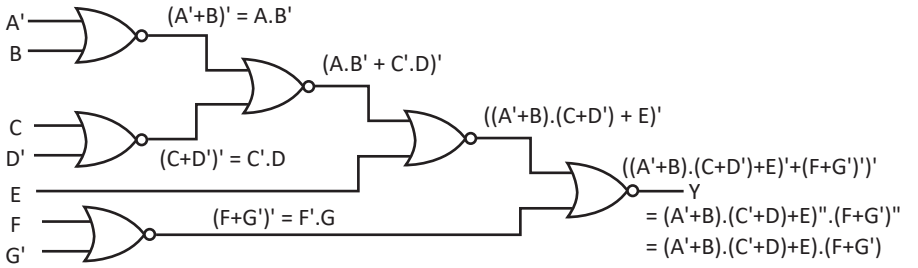


8.

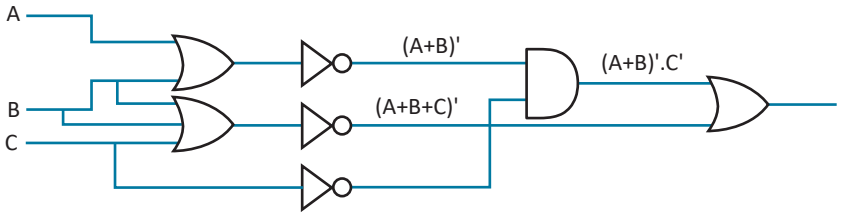


Answer: A.B' + A'.B

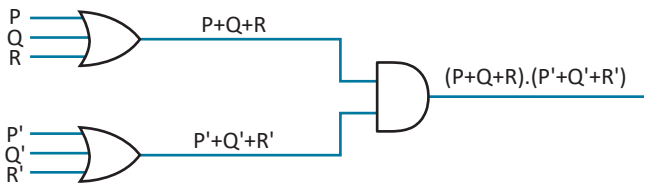
9.



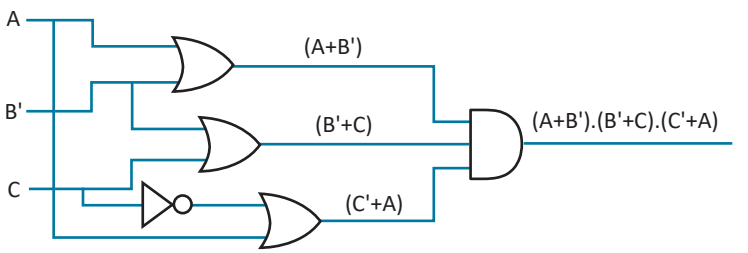
10. (i)



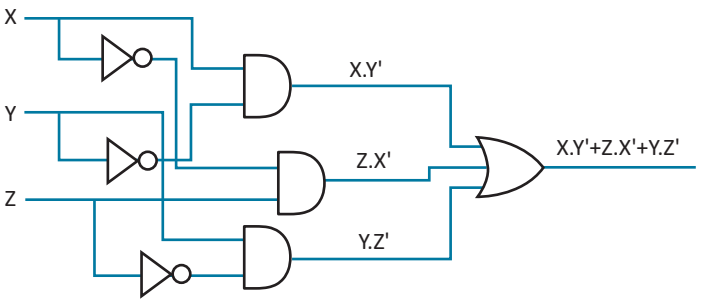
(ii)



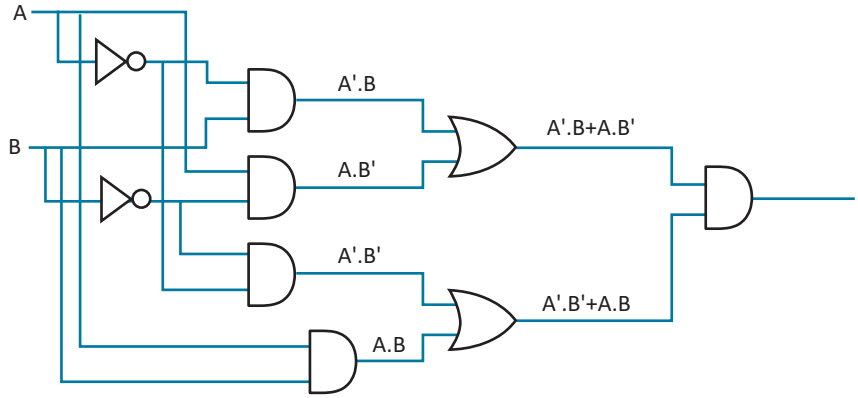
(iii)



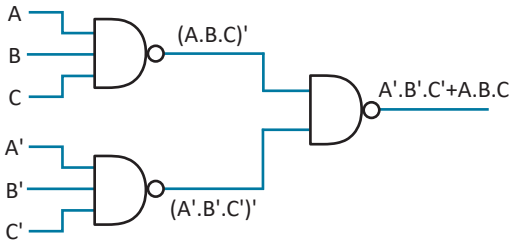
(iv)



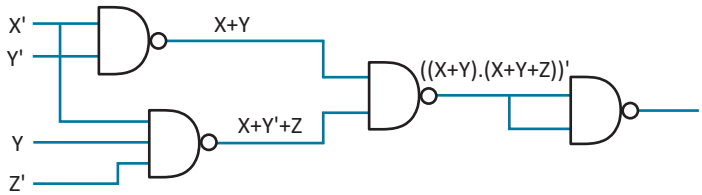
(v)



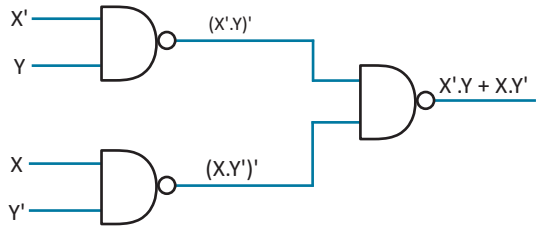
11. (i)



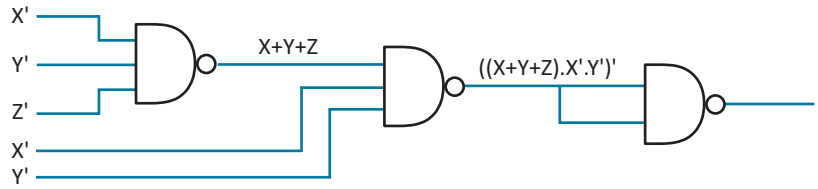
(ii)



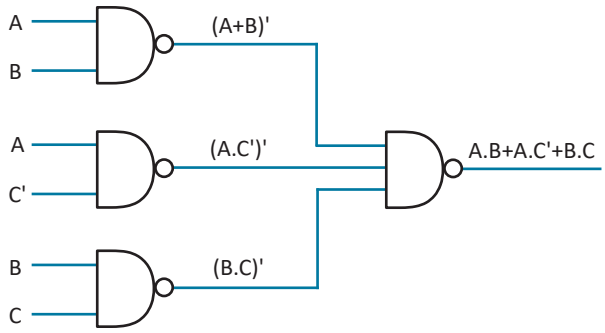
(iii)



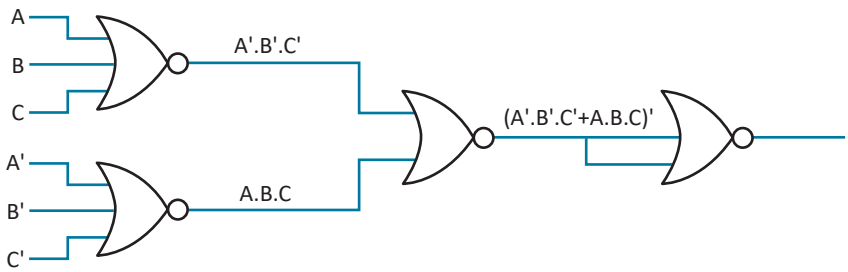
(iv)



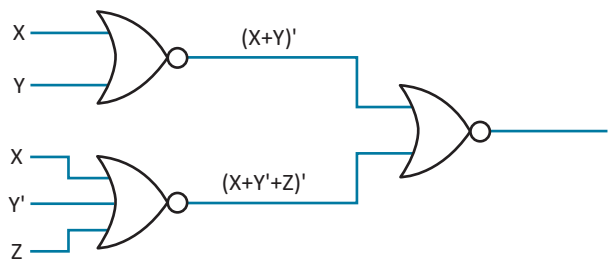
(v)



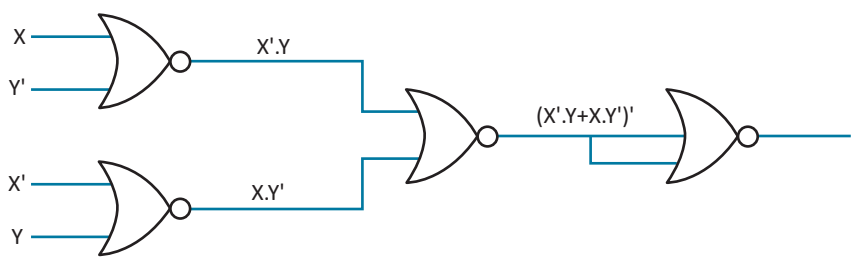
12. (i)



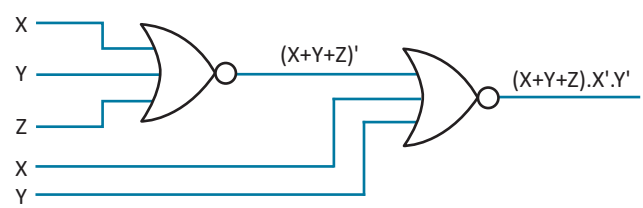
(ii)

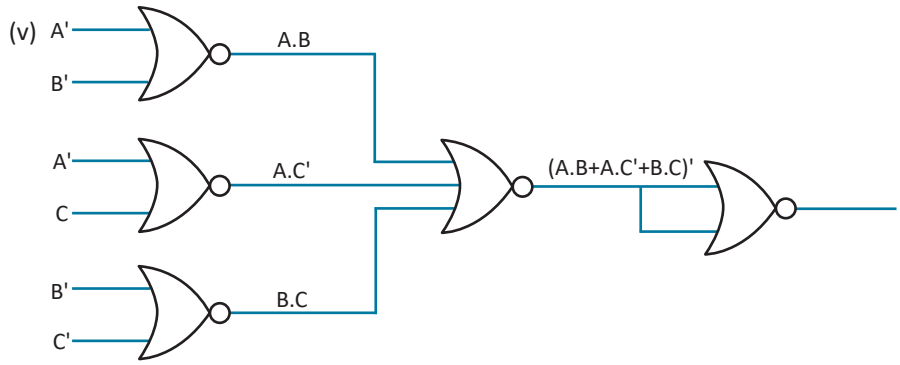


(iii)



(iv)





D. (c) Assertion is true and Reason is false.

3. Implementation of Algorithms to Solve Problems

Unsolved Questions

A. 1. d 2. c 3. c 4. b 5. b

B. 1. Flowchart 2. time 3. efficient 4. finite 5. Best

C. 1. a. **Algorithm:** An algorithm is a set of well-defined finite steps or rules to be followed to solve any given problem.

b. **Pseducode:** Pseudocode is a representation of an algorithm in any standard human readable language and mathematical notations.

c. **Flowchart:** A flowchart is a pictorial representation of an algorithm using standard symbols.

2. a. The oval-shaped start/stop symbol is used to represent the beginning/end of a flowchart.

b. The rectangle-shaped process box is used to represent arithmetic or assignment operations carried out internally.

c. The diamond-shaped decision box is used to check the condition. It has one input and two outputs representing the actions to be taken for two outcomes true and false.

d. The parallelogram-shaped input/output box represents an input operation like accepting data from the keyboard or an output operation like displaying the result on a monitor respectively.

e. The off-page connectors are used to join symbols on another page. These connectors are also used in pairs.

3. **Similarity between pseudo code and flowchart**

Both are used during the planning and design phase of a solution to help programmers and non-programmers understand how the logic will work before actual coding begins.

Difference between pseudo code and flowchart

Pseudo code: Uses natural language mixed with programming-like structures.

Flowchart: Visual representation using shapes (such as ovals, rectangles, and diamonds) connected by arrows to show the flow of steps.

4. a. **Best case complexity:** For any input of size 'n', the function takes the minimum time or the minimum number of steps for execution. It defines the lower bound of an algorithm.
   b. **Worst case complexity:** For any input of size 'n', the function takes the maximum time or the maximum number of steps for execution. It defines the upper bound of an algorithm.
   c. **Average case complexity:** In the average case analysis, we take all the possible inputs combination and calculate their individual computation time and then calculate the average of all the data. So, average case complexity calculates the average time taken by an algorithm for execution.
5. Follow the given steps:
   Step 1: Start
   Step 2: Input Cost Price (cp) and Selling Price (sp)
   Step 3: If sp > cp, then go to Step 4, else if sp < cp, go to Step 7, else go to Step 11
   Step 4: Calculate profit: p = sp - cp
   Step 5: Calculate profit percent: pp = (p / cp) * 100
   Step 6: Display profit (p) and profit percent (pp)
   Step 7: If sp < cp, then go to Step 8, else go to Step 11
   Step 8: Calculate loss: l = cp - sp
   Step 9: Calculate loss percent: lp = (l / cp) * 100
   Step 10: Display loss (l) and loss percent (lp)
   Step 11: Display "Neither profit nor loss"
   Step 12: Stop
6. Follow the given steps:
   Step 1: Start
   Step 2: Accept three numbers in a,b,c
   Step 3: If a<b and a<c then assign smallest s=a, go to Step 6
   Step 4: If b<a and b<c then assign smallest s=b, go to Step 6
   Step 5: If c<a and c<b then assign smallest s=c
   Step 6: Display s
   Step 7: Stop
7. Follow the given steps:
   Step 1: Start
   Step 2: Accept any number n
   Step 3: if n > 0 then display "positive number" , go to step 6
   Step 4: if n < 0 the display "negative number", go to step 6
   Step 5: Display "Zero"
   Step 6: Stop

8. Follow the given steps:

   Step1: Start

   Step 2: Accept three angles a,b,c

   Step 3: If a+b+c=180 then go to step 4 else go to step 7

   Step 4: If a<90 and b<90 and c<90 the print "acute angled triangle", go to step 8

   Step 5: If a>90 or b>90 or c>90 the print "obtuse angled triangle", go to step 8

   Step 6: If a=90 or b=90 or c=90 the print "right angled triangle", go to step 8

   Step 7: Print "triangle not possible"

   Step 8: Stop

9. Follow the given steps:

   Step1: Start

   Step 2: Accept three sides a,b,c

   Step 3: If a+b>c and b+c>a and c+a> b then go to step 4 else go to step 7

   Step 4: If a=b and b=c and c=a the print "equilateral triangle", go to step 8

   Step 5: If a!=b and b!=c and c!=a the print "scalene triangle", go to step 8

   Step 6: Print "isosceles triangle", go to step 8

   Step 7: Print " triangle not possible"

   Step 8: Stop

10. Follow the given steps:

    a. Decimal to Binary

       Step 1: Start

       Step 2: Input the decimal number n.

       Step 3: Initialise an empty string binary = "".

       Step 4: Repeat steps 5 to 7 until n > 0:

       Step 5: Find remainder = n % 2.

       Step 6: Append binary = remainder + binary.

       Step 7: Update n = n /2

       Step 8: If n = 0 set binary = "0".

       Step 9: Print binary

       Step 10: Stop

    b. Decimal to Octal

       Step 1: Start

       Step 2: Input the decimal number n.

       Step 3: Initialise an empty string octal = "".

       Step 4: Repeat steps 5 to 7 until n > 0:

       Step 5: Find remainder = n % 8

       Step 6: Append octal= remainder + octal

Step 7: Update n = n /8

Step 8: If n = 0 set octal = "0".

Step 9: Print octal

Step 10: Stop

c. Decimal to Hexadecimal

Step 1: Start

Step 2: Input the decimal number n.

Step 3: Initialise an empty string hexa = "".

Step 4: Repeat steps 5 to 7 until n > 0:

Step 5: Find remainder = n % 16

Step 6: If remainder <=9 then append hexa= remainder + hexa else append hexa = (character) (remainder+55) +hexa

Step 7: Update n = n /8

Step 8: If n = 0 set hexa= "0".

Step 9: Print hexa

Step 10: Stop

11. Follow the given steps:

Step 1: Start

Step 2: Initialise variables term1 to 1, term2 to 2, term3 to 0 and i to 1

Step 3: Read number of terms in num

Step 4: Display term1 and term2

Step 5: Repeat Step 5 to Step 10 until i <= num -2

Step 6: Calculate term3= term1+2*term2

Step 7: Display term3

Step 8: Initialise term1 = term2

Step 9: Initialise term2 = term3

Step 10: Increment i by 1

Step 11: Stop

12. Follow the given steps:

Step 1: Start

Step 2: Accept number n

Step 3: Initialise f=2

Step 4: Repeat step 5 to step 8 until n=1

Step 5: If n%f=0 then go to step 6 else go to step

Step 6: Display f

Step 7: Assign n=n/f, go to step 4

Step 8: Increment f by 1

Step 9: Stop

13. Follow the given steps:

(Assuming the array is already created and the variables already entered)

Step 1: Start

Step 2: Initialise loop variable i to 0 and index position pos to -1

Step 3: Repeat step 4 to step 6 until i <n

Step 4: if a[i] = s then go to step 5

else go to step 6

Step 5: Initialise pos to i, go to step 7

Step 6: Increment i by 1

Step 7: if pos!= -1 then go to step 8 else go to step 9

Step 8: Replace a[p] with n , go to step 10

Step 9: Display s "not found"

Step 10: Stop

14. Follow the given steps:

Step 1: Start

Step 2: Initialise i to 0, sum to 0 and standard deviation sd to 0

Step 3: Repeat step 4 to step 5 while i<n

Step 4: Add a[i] to sum

Step 5: Increment i by 1

Step 6: Calculate mean=sum/n

Step 7: Initialise i to 0 and variance to 0

Step 8: Repeat step 9 to step 10 while i<n

Step 9: Add (a[i]-mean)2 to variance

Step 10: Increment i by 1

Step 11: Calculate sd= squareroot(variance /n)

Step 12: Display sd

Step 13: Stop

15. Follow the given steps:

Step 1: Start

Step 2: Intitialise variable i to 0

Step 3: Repeat step 4 to step 10 until i<9

Step 4: Intitialise variable j to 0

Step 5: Repeat step 6 to step 9 until j <9- i -1

Step 6: If temp[j] < temp[j+1] then go to step 7 else go to step 9

Step 7: Swap temp[j] and temp[j+1]

Step 8: Swap city[j] and city[j+1]

Step 9: Increment j by 1

Step 10: Increment i by 1

Step 11: Stop

16. Follow the given steps:
    Step 1: Start
    Step 2: Declare matrix mirror[r][c]
    Step 3: Initialise i to 0
    Step 4: Repeat step 5 to step 12 until i< r
    Step 5: Initialise x to c-1
    Step 6: Initialise j to 0
    Step 7: Repeat step 8 to step 10 until j<c
    Step 8: Assign mirror[i][x]=a[i][j]
    Step 9: Decrease x by 1
    Step 10: Increment j by 1
    Step 11: Increment i by 1
    Step 12: Display matrix mirror[][]
    Step 13: Stop
17. Follow the given steps:
    Step 1: Start
    Step 2: Initialise flag to true
    Step 3: Initialise i to 0
    Step 4: Initialise j to 0
    Step 5: If i=j and a[i][j] !=1 the assign flag=false, go to step 9
    Step 6: If i!=j and a[i][j]!=0 then assign flag=false, go to step 9
    Step 7: Increment j by 1
    Step 8: Increment i by 1
    Step 9: if flag=true then display "Unit Matrix", go to step 11
    Step 10: Display "Not Unit Matrix"
    Step 11: Stop
18. Follow the given steps:
    Step 1: Start
    Step2: Accept any sentence in str
    Step 3: Create a StringTokenizer object st with delimeters space, full stop, comma, question mark
    Step 4: Repeat step 5 to step 6 until st.hasMoreTokens() returns false
    Step 5: Extract word=st.nextToken()
    Step 6: If word.charAt(0) is 'A' or 'E' or 'I' or 'O' or 'U' or 'a' or 'e' or 'i' or 'o' or 'u' then display word
    Step 7: Stop
19. Follow the given steps:
    Step 1: Start
    Step2: Accept any word in upper case and store in wrd

Step 3: Initialise sorted string swrd to null and length len=wrd.length()

Step 4: Initialise i to 65

Step 5: Repeat step 6 to step 11 until i=90

Step 6: Initialise j to 0

Step 7: Repeat step 8 to step 10 until j=len

Step 8: Extract character ch from position j

Step 9: if ch = (char) i the append ch to swrd

Step 10: Increment j by 1

Step 11: Increment i by 1

Step 12: Display swrd

Step 13: Stop

20. Follow the given steps:

Step 1: Start

Step2: Accept any word in upper case and store in wrd

Step 3: Initialise reverse string rwrd to null and length len=wrd.length()

Step 4: Initialise i to 0

Step 5: Repeat step 6 to step 8 until i=len

Step 6: Extract character ch from position j

Step 7: Assign rwrd=rwrd+ch

Step 8: Increment i by 1

Step 9: If wrd = rwrd then display "palindrome", go to step 13

Step 12: Display "not palindrome"

Step 13: Stop

21. Follow the given steps:

Step 1: Start

Step 2: if number= 0 then return high

Step 3: Assign digit=number%10

Step 4: if high<digit then return highest(number/10,digit) else Call highest(number/10,high) and goto step 2

Step 5: Stop

22. Follow the given steps:

Step 1: Start

Step 2: If binary=0 then return deci

Step 3: Add binary%10 position to previous value of deci

Step 4: Call decimal(binary/10, position+1) and go to step 2

Step 5: Stop

23. Follow the given steps:

    Step 1: Start

    Step 2: if number=0 return true and go to step 5

    Step 3: If number%10 != square%10 return false and go to step 5

    Step 4: Return automorphic(number/10,square/10) and go to step 2

    Step 5: Stop

24. Follow the given steps:

    Step 1: Start

    Step 2: Create new nodes ptr and temp

    Step 3: Assign ptr.data=item and ptr.next=null

    Step 4: Assign temp=start

    Step 5: Initialise c to 1

    Step 6: Repeat Step 7 to Step 8 while c<p-1

    Step 7: Assign temp = temp.next

    Step 8: Increase c by 1

    Step 9: Assign ptr.next= temp.next

    Step 10: Set temp.next= ptr

    Step 11: Stop

25. Follow the given steps:

    Push operation of stack

    Step 1: Start

    Step 2: Create a new node ptr

    Step 3: Assign value of that node as ptr->data = value

    Step 4: Assign node ptr->link as header->link

    Step 5: Assign header=ptr

    Step 6: Stop

    Pop operation of stack

    Step 1: Start

    Step 2: If node=null then display "Underflow"

    Step 3: Initialise pointer temp to the top node and move forward the top node by 1 as ptr = ptr->link

    Step 4: Initialise temp->data=0 and temp->link=null

    Step 5: Stop

26. Follow the given steps:

    Insert operation in Queue

    Step 1: Start

    Step 2: Create a node ptr

    Step 3: Assign ptr->data=value

Step 4: if front=null then go to step 5 else go to step 7

Step 5: Assign front=ptr and rear=ptr

Step 6: Assign front->link and rear-> as null , go to step 9

Step 7: Assign rear->link=ptr

Step 8: Assign rear=ptr and rear->next=null

Step 9: Stop

Deletion operation in Queue

Step 1: Start

Step 2: Create a node ptr

Step 3: if front=null then display "underflow" , go to step 7

Step 4: Assign ptr=front

Step 5: Assign front=front->link

Step 6: Assign ptr=null

Step 7: Stop

27. Follow the given steps:

Step 1: Start

Step 2: Create new nodes ptr1 and head2

Step 3: Initialise c =1

Step 4: Initialise ptr1=head1

Step 5: Repeat step 6 to Step 7 until c=p

Step 6: Move pointer to next node as ptr1=ptr1>link

Step 7: Increment c by 1

Step 8: Assign head2=ptr1

Step 9: Assign ptr->link = null

Step 10: Stop

28. Follow the given steps:

Step 1: Start

Step 2: Create nodes previous, next and initialise it to null

Step 3: Create node current and initialise it to head

Step 4: Repeat step 4 to step 7 until current->link=null

Step 5: Assign next=current->link

Step 6: Assign current->link=previous

Step 7: Assign previous=current and current=next

Step 8: Stop

D. (d) Assertion is false and Reason is true.

4. Programming in Java

Unsolved Questions

A. 1. a 2. a 3. b 4. c 5. d
6. d 7. a 8. a 9. b 10. c

B. 1. Procedure Oriented Language
2. Single line
3. JAVA DEVELOPMENT KIT
4. Write Once, Run Anywhere
5. ob

C. 1. There are some drawbacks of procedure-oriented programming language. They are as follows:
- Different data values work on the same functions, so there should be some necessary changes in the function so that the code works properly.
- Real situations cannot be handled properly as they require complex programming.
- Its code is often not reusable.
- It provides less security as the data is exposed to the whole program.

2. Examples of Superclass : shape and Animal is the superclass
Example of subclass : rectangle and Dog is the subclass

3. The advantages of object-oriented programming languages are as follows:
- Each and every instance has properties that are exclusively for its own and cannot be used by other instances. For example, if you have a Fan's blueprint, you can create many instances of fans with different model names such as Crompton Greaves, Usha, Cinni, etc. Thus, a real-world picture can be seen.
- The improved quality of software and reduced cost of development are its key features as it uses the concept of reusability. Reusability allows the code to be written once and used multiple times.
- Maintaining and modifying existing code is much easy when you try to create new objects with small differences.
- Abstraction and Data hiding maintain the security of data as only necessary data is provided.
- We can eliminate the redundant code using the concept of Inheritance. We can write common class definitions for similar functionalities and inherit them in a different class. Hence, data redundancy is decreased.

4. Class is a blueprint or prototype that is required to create objects of the same kind. Thus, a class is said to be a collection of those objects that have the same characteristics and behaviours.
An object is a unique entity that contains properties, methods and events together in an Object-Oriented Programming language.

5. An object is an instance of a class because it is created based on the class's blueprint, which defines its structure and behavior. The class provides the template, and the object is a specific manifestation of that template.
6. Data hiding is a part of encapsulation, which combines data and methods into a single class. By using encapsulation, you can expose only the necessary parts of an object's functionality through public methods while keeping the internal data and implementation hidden. This helps in maintaining control over how the data is accessed and modified.
7. Data Abstraction is the property by which the essential features are represented without knowing the background details that how it is actually executing, i.e., non-essential units are hidden from the user.
8. Encapsulation is a procedure of combining data and functions together into a single unit or entity.
9. `class_name object_name = new constructor();`

   eg. If the name of the class is Car and the object is Hundai, then the object is created in the following way:

   ```
   Car Hundai = new Car();
   ```

10.

POP Language	OOP Language
Procedural Programming follows the top-down approach.	Object-Oriented Programming follows the bottom-up approach.
Code is often not reusable.	Reusability allows the code to be written once and used multiple times.

5. Primitive Values, Wrapper Classes, Types and Casting

Unsolved Questions

A. 1. a 2. d 3. a 4. d 5. d

B. 1. boolean 2. Integer 3. eight 4. explicit conversion 5. impure

C. 1. This process of converting a lower data type into a higher data type is known as widening.

2. Autoboxing is a method of converting a primitive data type into an object of its corresponding wrapper class. For example, an int data type variable is converted into its object by wrapping the value with the wrapper class "Integer".
3. To write codes in Java, different characters such as alphabets, digits and special characters are used. The different combinations of these characters are used to declare variables, keywords and symbols which are required to write a whole program. These are known as Character Sets of Java. They are also called Unicode Character Sets.
4. Unboxing is a method of converting an object of a wrapper class into its corresponding primitive data type value. For example, an object of the Double class is converted into a double variable.

For example, converting Double class into double:

```
double d_object = new Double(267.5689);
double d_datatype = d_object;
```

5.

Primitive Data Type	Non-Primitive Data Type
Primitive data types in Java are predefined by the language and directly store actual values in memory.	Non-primitive data types are user-defined and store references to memory locations where the actual data.
Primitive data types have fixed sizes defined by the Java specification.	Non-primitive data types have variable sizes based on the complexity of the data.
Primitive data types are immutable, meaning their values cannot be altered once assigned, and are compared by their actual values.	Non-primitive data types are mutable, allowing their internal state to be changed, and are compared based on their reference addresses in memory.

6. A Wrapper class in Java is a type of class that is used to convert primitive data types into objects and objects to primitive data types. It wraps around a variable of one data type and converts it into an object.

D. (a) Both Assertion and Reason are true, and Reason is the correct explanation for Assertion.

6. Variables and Expressions

Unsolved Questions

A.

1. b	2. a	3. b	4. b	5. b
6. a	7. b	8. b	9. b	10. c
11. c	12. c	13. b	14. c	

B.

1. true	2. relational	3. Left to Right	4. Shorthand	5. <<
6. high	7. four	8. true	9. five	10. false

C.

1. Unary, Binary and Ternary
2. Operators that works on two operands. Eg. A+B
3. This operator contains only one operand. For example: a++, --b
   The different types of Unary Operators are : Unary +, Unary - , Increment ++, Decrement --
4.
   - 5
     4
   - 20
     20
   - 10
     4
   - 278

5. The logical operators are the symbols or words that connect two or more relational expressions. They determine the logic between the variables or values. Their output is the outcome of different conditions given and they always return "true" or "false" depending on the relationship between the parameters of the expression.

   There are 3 logical operators in Java : AND (&&), OR (||) and NOT(!)

6. A variable is assigned some value to perform some task in a program. This is called the assignment statement and the symbol '=' used in this statement is known as an assignment operator. The left of the assignment operator is a variable that stores the value that is on the right side of the assignment operator.

   Eg s=a+b where = is the assignment operators.

7. x = 50

   y = -70

   z = -175000

8. 4 0 0

9. 1.024390243902439

10. Bitwise Operators work on the bit level of the operands on which the expression depends. The operands are of Integer Literals (i.e. byte, int, short and long). It cannot work on Real Literals.

11. 520

12. Greatest number: 10

13. You are eligible to vote

D. (b) Both Assertion and Reason are true, but Reason is not the correct explanation for Assertion.

7. Statements and Scope

Unsolved Questions

A.

1. b	2. a	3. b	4. b	5. a
6. a	7. b	8. a	9. c	10. d

B.

1. Conditional operator
2. 5
3. Marks=10
4. 2

   4

   5
5. CAT
6. 15

7. Continue
8. 2
9. BYE
10. switch

C. 1.
```
int gcd(int a, int b)
{
    int r;
     while(b != 0) // ?1? condition to continue the loop until b becomes 0
    {
        r = a % b;  // ?2? r is the remainder of a divided by b
        a = b;      // ?3? a is updated to the value of b
        b = r;      // ?4? b is updated to the remainder
    }
    if(a == 0)
        return b;   // ?5? return b as the GCD
    else
        return -1;
}
```

2.
```
for(int a = 50; a >= 1; a--) {
    System.out.println(a);
}
```

3. 1,1
   2,1 2,2
   3,1 3,2 3,3
   4,1 4,2 4,3 4,4
4. 2
   5
   :
   A
   J
5.
```
if(x >= 'A' && x <= 'Z') {
    c = "Uppercase";
} else {
    c = "Lowercase";
}
```

6. System.out.println((a >= b && a < c) ? (a + b) : (a * b));
7. TRUE
8. 6
9. 18
10.
```
int i=10;
while(i>=1)
{
System.out.println(i*10);
i=i-1;
}
```
11.
```
import java.util.*;
class Numbers {
    // Data member
    int n;
    // Member function to accept integer n
    void getnum() {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        n = sc.nextInt();
    }
    // Member function to check if n is triangular
    int check(int n) {
        int sum = 0;
        for (int i = 1; sum < n; i++) {
            sum += i;
            if (sum == n) {
                return 1; // Triangular number
            }
        }
        return 0; // Not a triangular number
    }

    // Member function to display the result
    void dispnum() {
```

```
            if (check(n) == 1) {
                System.out.println(n + " is a triangular number.");
            } else {
                System.out.println(n + " is not a triangular number.");
            }
        }
        public static void main(String[] args)
        {
            Numbers ob= new Numbers();
            ob.getnum();
            ob.dispnum();
        }
    }
```

12.
```
    import java.util.*;
    class Special
    {
        // Data member
        int n;
        // Default constructor
        Special() {
            n = 0;
        }
        // Parameterized constructor
        Special(int value) {
            n = value;
        }
        // Method to calculate and display the sum of the first and last
        digit of n
        void sum() {
            // Convert the number to a string to easily access digits
            int t=n,s=0,len,fd,ld;
            len=Integer.toString(n).length();
            ld=t%10;
            fd=t/(int)Math.pow(10,len-1);
```

```
        s=ld+fd;
       System.out.println("Sum of the first and last digit: " + s);
    }
    // Method to check if n is a special number
    void isSpecial()
    {
        int temp = n;
        int sumOfFactorials = 0;
        while(temp>0)
        {
            int f=1,i;
            for(i=1;i<=temp%10;i++)
            {
                f=f*i;
            }
            sumOfFactorials += f;
            temp=temp/10;
        }
        // Check if the number is a special number
        if (sumOfFactorials == n) {
            System.out.println(n + " is a special number.");
        } else {
            System.out.println(n + " is not a special number.");
        }
    }
    public static void main(String[] args)
    {
        Scanner sc= new Scanner(System.in);
        int n;
        System.out.print("Enter a number : ");
        n=sc.nextInt();
        Special ob = new Special(n);
        ob.sum();
        ob.isSpecial();
```

```
        }
    }
```

13.
```
import java.util.*;
class Taximeter
{
    // Data members
    int taxino;
    String name;
    int km;
    double billAmount;
    // Default constructor
    Taximeter()
    {
        taxino = 0;
        name = "";
        km = 0;
        billAmount = 0.0;
    }
    // Method to input data
    void input(int taxino, String name, int km)
    {
        this.taxino = taxino;
        this.name = name;
        this.km = km;
       calculate(); // Calculate the bill amount when data is input
    }
    // Method to calculate bill amount
    void calculate()
    {
        if (km <= 1) {
            billAmount = 25;
        } else if (km <= 6) {
            billAmount = 25 + (km - 1) * 10;
        } else if (km <= 12) {
```

```
            billAmount = 25 + 5 * 10 + (km - 6) * 15;
        } else if (km <= 18) {
            billAmount = 25 + 5 * 10 + 6 * 15 + (km - 12) * 20;
        } else {
            billAmount = 25 + 5 * 10 + 6 * 15 + 6 * 20 + (km - 18)
* 25;
        }
    }
    // Method to display the details
    void display() {
        System.out.println("Taxino"+"\t"+ "Name"+"\t"+" Kilometres
        travelled"+"\t"+" Bill amount");
        System.out.println( taxino+"\t"+ name+"\t"+ km+"\t"+
        billAmount);
    }
    public static void main(String[] args)
    {
        Taximeter ob= new Taximeter();
        ob.input(2345,"Rupesh",68);
        ob.calculate();
        ob.display();
    }
}
```

14.
```
class TelephoneBill {
    // Data members
    long phoneNumber;
    String name;
    final int HIRE_CHARGE = 200; // constant hire charge
    int unitsConsumed;
    float billAmount;
   // Constructor to initialize phoneNumber, name, and unitsConsumed
    public TelephoneBill(long phoneNumber, String name, int
    unitsConsumed) {
        this.phoneNumber = phoneNumber;
        this.name = name;
```

```
        this.unitsConsumed = unitsConsumed;
        this.billAmount = 0; // billAmount is calculated later
    }
    // Method to calculate the bill amount
    public void calculateBill() {
        float unitCharge = 0;
        if (unitsConsumed <= 100) {
            unitCharge = unitsConsumed * 1.0f;
        } else if (unitsConsumed <= 200) {
            unitCharge = (100 * 1.0f) + ((unitsConsumed - 100) *
            1.5f);
        } else {
            unitCharge = (100 * 1.0f) + (100 * 1.5f) + ((unitsConsumed
            - 200) * 2.0f);
        }
        billAmount = HIRE_CHARGE + unitCharge;

    }
    // Method to display the bill for the subscriber
    public void displayBill() {
        System.out.println("Subscriber Name: " + name);
        System.out.println("Phone Number: " + phoneNumber);
        System.out.println("Units Consumed: " + unitsConsumed);
        System.out.println("Hire Charge: " + HIRE_CHARGE);
        System.out.println("Bill Amount: " + billAmount);
    }
    public static void main(String[] args)
    {
        // Example usage
        TelephoneBill subscriber = new TelephoneBill(9876543210L,
        "John Doe", 220);
        subscriber.calculateBill();
        subscriber.displayBill();
    }
}
```

15.
```
import java.util.*;

class PhoneBill
 {
    // Data members
    char[] customerName = new char[50];
    long phoneNumber;
    int no_of_units;  // Number of calls
    int rent;
    float amount;
    // Constructor to assign initial values
    PhoneBill()
    {
        phoneNumber = 0L;
        no_of_units = 0;
        rent = 0;
        amount = 0.0f;
    }
    // Member function to accept values and calculate the bill
    void accept()
    {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter customer name: ");
        String name = sc.nextLine();
        customerName = name.toCharArray();
        System.out.print("Enter phone number: ");
        phoneNumber = sc.nextLong();
        System.out.print("Enter number of calls (units): ");
        no_of_units = sc.nextInt();
        System.out.print("Enter rent: ");
        rent = sc.nextInt();
        // Calculate the bill amount
        calculate();
    }
```

```
    // Member function to calculate the bill amount
    void calculate()
    {
        float cost = 0;
        if (no_of_units > 50) {
            if (no_of_units <= 150) {
                cost = (no_of_units - 50) * 0.80f;
            } else if (no_of_units <= 350) {
               cost = (100 * 0.80f) + ((no_of_units - 150) * 1.0f);
            } else {
                cost = (100 * 0.80f) + (200 * 1.0f) + ((no_of_units
                - 350) * 1.20f);
            }
        }
        // Total amount is rent plus calculated cost
        amount = rent + cost;
    }
    // Member function to display the bill details
    void display()
    {
      System.out.println("Customer Name: " + new String(customerName));
        System.out.println("Phone Number: " + phoneNumber);
        System.out.println("Number of Calls: " + no_of_units);
        System.out.println("Rent: " + rent);
        System.out.println("Total Bill Amount: " + amount);
    }
    public static void main(String[] args)
    {
        // Example usage
        PhoneBill pb = new PhoneBill();
        pb.accept();
        pb.display();
    }
}
```

16.
```
import java.util.Scanner;
class PrimeDigits {
    // Data member
    int n;
    // Parameterized constructor to initialize the value of n
    public PrimeDigits(int n) {
        this.n = n;
    }
    // Function to check if a digit is prime
    public boolean isPrime(int digit) {
        if (digit <= 1) {
            return false;
        }
        for (int i = 2; i <= Math.sqrt(digit); i++) {
            if (digit % i == 0) {
                return false;
            }
        }
        return true;
    }
    // Function to calculate the sum of prime digits
    public int sumOfPrimeDigits() {
        int sum = 0;
        int num = n;
        // Loop through each digit of the number
        while (num > 0) {
            int digit = num % 10;
            if (isPrime(digit)) {
                sum += digit;
            }
            num /= 10; // Remove the last digit
        }
        return sum;
    }
```

```
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int n = sc.nextInt();
        PrimeDigits pd = new PrimeDigits(n);
     System.out.println("Sum of prime digits: " + pd.sumOfPrimeDigits());
    }
}
```

17.
```
import java.util.Scanner;
public class SeriesCalculator {
      // Helper function to calculate factorial
    public static long factorial(int num) {
        long fact = 1;
        for (int i = 2; i <= num; i++) {
            fact *= i;
        }
        return fact;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        // Display the menu
        System.out.println("Choose the series to calculate:");
        System.out.println("1. x + x/2! + x/3! + x/4! + ... + x/n!");
        System.out.println("2. x/2! + x^2/3! + x^3/4! + ... + x^n/
        (n+1)!");
        System.out.println("3. x/2! - x^2/3! + x^3/4! - x^4/5! ±
        x^n/(n+1)!");
        int choice = sc.nextInt();
        // Prompt user for x and n
        System.out.print("Enter the value of x: ");
        double x = sc.nextDouble();
        System.out.print("Enter the value of n: ");
        int n = sc.nextInt();
        switch (choice) {
```

```
                case 1:
                    // Case 1: Sum = x + x/2! + x/3! + ... + x/n!
                    double sum1 = x;
                    for (int i = 2; i <= n; i++) {
                        sum1 += x / factorial(i);
                    }
                    System.out.println("Sum of the series: " + sum1);
                    break;
                case 2:
                    // Case 2: Sum = x/2! + x^2/3! + x^3/4! + ... + x^n/
                    (n+1)!
                    double sum2 = 0;
                    for (int i = 2; i <= n + 1; i++) {
                        sum2 += Math.pow(x, i - 1) / factorial(i);
                    }
                    System.out.println("Sum of the series: " + sum2);
                    break;
                case 3:
                    // Case 3: Sum = x/2! - x^2/3! + x^3/4! - x^4/5! ±
                    x^n/(n+1)!
                    double sum3 = 0;
                    int sign = 1;
                    for (int i = 2; i <= n + 1; i++) {
                     sum3 += sign * (Math.pow(x, i - 1) / factorial(i));
                        sign *= -1;  // Alternate the sign
                    }
                    System.out.println("Sum of the series: " + sum3);
                    break;
                default:
                    System.out.println("Invalid choice!");
                    break;
            }
            sc.close();
        }
    }
```

18.
```
import java.util.Scanner;
public class SeriesSum {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        // Display the menu
        System.out.println("Choose the series to calculate:");
        System.out.println("1. Sum (S) = 2 - 4 + 6 - 8 + ……. -20");
        System.out.println("2. Sum (S) = x/2 + x/5 + x/8 + x/11 + … + x/20");
        int choice = sc.nextInt();
        switch (choice) {
            case 1:
                // Case 1: Sum (S) = 2 - 4 + 6 - 8 + ……. -20
                int sum1 = 0;
                for (int i = 2; i <= 20; i += 2) {
                    sum1 += (i % 4 == 0) ? -i : i;
                }
                System.out.println("Sum of the series: " + sum1);
                break;
            case 2:
                // Case 2: Sum (S) = x/2 + x/5 + x/8 + x/11 + … + x/20
                System.out.print("Enter the value of x: ");
                double x = sc.nextDouble();
                double sum2 = 0;
                for (int i = 2; i <= 20; i += 3) {
                    sum2 += x / i;
                }
                System.out.println("Sum of the series: " + sum2);
                break;
            default:
                System.out.println("Invalid choice!");
                break;
        }
```

```
            sc.close();
        }
    }
```

19.
```
import java.util.Scanner;
public class TrianglePattern {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        // Asking for the type of triangle
        System.out.print("Enter 1 for a triangle or 2 for an inverted
        triangle: ");
        int type = sc.nextInt();
        // Asking for the number of terms
        System.out.print("Enter the number of terms: ");
        int n = sc.nextInt();
        switch (type) {
            case 1:
                // Generate a normal triangle pattern
                for (int i = 1; i <= n; i++) {
                    for (int j = 1; j <= i; j++) {
                        System.out.print(i);
                    }
                    System.out.println();
                }
                break;
            case 2:
                // Generate an inverted triangle pattern
                for (int i = n; i >= 1; i--) {
                    for (int j = 1; j <= i; j++) {
                        System.out.print(i);
                    }
                    System.out.println();
                }
                break;
            default:
```

```
                System.out.println("Invalid choice! Please enter 1
                or 2.");
                break;
            }
            sc.close();
        }
    }
```

20.
```
import java.util.*;
public class SmithNumber {
    // Function to calculate the sum of digits of a number
    public static int sumOfDigits(int num) {
        int sum = 0;
        while (num > 0) {
            sum += num % 10;
            num /= 10;
        }
        return sum;
    }
    // Function to check if a number is prime
    public static boolean isPrime(int num) {
        if (num < 2) {
            return false;
        }
        for (int i = 2; i <= Math.sqrt(num); i++) {
            if (num % i == 0) {
                return false;
            }
        }
        return true;
    }
    // Function to find and return the sum of digits of the prime
    factors of a number
    public static int sumOfPrimeFactorsDigits(int num) {
        int sum = 0;
```

```
        int factor = 2;
        // Finding prime factors
        while (num > 1) {
            while (num % factor == 0) {
                sum += sumOfDigits(factor); // Add the sum of digits
                of the prime factor
                num /= factor;
            }
            factor++;
        }
        return sum;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int number = sc.nextInt();
        // If the number is prime, it cannot be a Smith number
        if (isPrime(number)) {
            System.out.println(number + " is not a Smith number
            (prime numbers are not considered Smith numbers).");
        } else {
            // Calculate sum of digits of the number
            int digitSum = sumOfDigits(number);
          // Calculate sum of digits of prime factors of the number
            int primeFactorDigitSum = sumOfPrimeFactorsDigits(numb
            er);
            // Check if it's a Smith number
            if (digitSum == primeFactorDigitSum) {
                System.out.println(number + " is a Smith number.");
            } else {
             System.out.println(number + " is not a Smith number.");
            }
        }
    }
}
```

21.
```
import java.util.Scanner;
public class SpyNumber
 {
    // Function to check if the number is a Spy Number
    public static boolean isSpyNumber(int num)
    {
        int sum = 0;
        int product = 1;
        int digit;
        // Loop through each digit of the number
        while (num > 0) {
            digit = num % 10;  // Extract the last digit
            sum += digit;      // Add it to the sum
            product *= digit;   // Multiply it to the product
            num /= 10;          // Remove the last digit
        }
        // Check if sum equals product
        return sum == product;
    }
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        // Input number
        System.out.print("Enter a number: ");
        int number = sc.nextInt();
        // Check if the number is a spy number
        if (isSpyNumber(number))
        {
            System.out.println(number + " is a Spy Number.");
        }
        else
        {
            System.out.println(number + " is not a Spy Number.");
        }
    }
}
```

22.
```
import java.util.*;
class Flight {
    // Instance variables
    int fl_no;
    String dest;
    float dist,fuel;
    // Method to calculate the fuel based on the distance
    public void calfuel()
    {
        if (dist <= 1000)
        {
            fuel = 500;
        }
        else if (dist > 1000 && dist <= 2000)
        {
            fuel = 1100;
        }
        else
        {
            fuel = 2200;
        }
    }
    // Method to input flight details
    public void feedinfo()
    {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter Flight Number: ");
        fl_no = sc.nextInt();
        sc.nextLine(); // Consume the newline left-over
        System.out.print("Enter Destination: ");
        dest = sc.nextLine();
        System.out.print("Enter Distance: ");
        dist = sc.nextFloat();
        // Calculate fuel based on distance
        calfuel();
```

```
    }
    // Method to display flight details
    public void showinfo()
    {
        System.out.println("Flight Number: " + fl_no);
        System.out.println("Destination: " + dest);
        System.out.println("Distance: " + dist + " km");
        System.out.println("Fuel Required: " + fuel + " liters");
    }
    public static void main(String[] args)
    {
        // Create a Flight object
        Flight flight = new Flight();
        // Input flight details
        flight.feedinfo();
        // Display flight details
        flight.showinfo();
    }
}
```

23.
```
import java.util.*;
class DeciOct {
    // Data Members
    private int n; // Stores the decimal number
    private String oct; // Stores the equivalent octal number
    // Constructor to initialize data members to 0
    public DeciOct() {
        n = 0;
        oct = "0";
    }
    // Method to assign a decimal number to n
    public void getnum(int nn) {
        n = nn;
    }
    // Method to calculate the octal equivalent of n and store it
    in oct
```

```
    public void deci_oct() {
        oct = Integer.toOctalString(n);
    }
   // Method to display the decimal number and its octal equivalent
    public void show() {
      // Call the deci_oct() method to calculate the octal equivalent
        deci_oct();
        // Display decimal and octal equivalents
        System.out.println("Decimal Number: " + n);
        System.out.println("Octal Equivalent: " + oct);
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a decimal number: ");
        int n = sc.nextInt();
        // Create an object of the DeciOct class
        DeciOct deciOct = new DeciOct();
        // Input decimal number
        deciOct.getnum(n);
        // Display decimal and octal equivalents
        deciOct.show();
    }
}
```

8. Methods

Unsolved Questions

A. 1. d 2. a 3. b 4. c 5. c
6. b 7. b 8. a 9. a 10. b

B. 1. Data types 2. return 3. parameters 4. formal 5. pass by value
6. methods 7. area of the circle 8. Method / Function overloading
9. basic 10. private

C.

1. The first line of any method definition is known as the method header or method prototype. It consists of the access specifier, return data type, method name and parameter list. eg: public int calculateSum(int a, int b)
2. A parameterised constructor in object-oriented programming is a type of constructor that allows you to initialise an object's fields with specific values at the time of its creation. Unlike a default constructor, which does not take any arguments and initialises fields with default values, a parameterised constructor accepts arguments that provide initial values for the object's attributes.
3. This method compares the given string in the parameter with the string in the current object alphabetically. It returns an integer value, i.e., the difference between the ASCII codes of the characters that are compared. If both strings are equal, it returns 0. If the first string is larger lexicographically than the second string, it returns a positive number else returns a negative value.
4. This method joins two strings together. The string of the current String object whose method is called and the string in the parameter are the two strings that are joined. This method returns a string.

   String st1 = "Computer";

   String st2 = "Desktop";

   String joinst = st1.concat(st2);
5. e
6. 4.5
7. The index value is :16
8. if(Character.isUpperCase(ch)) snippet is used to determine if the character ch is an uppercase letter and allows you to execute code conditionally based on this check.
9. The snippet computes the square root of 25 and 3 raised to the power of 2, then returns the larger of the two results. In this case, the result is 9.0.
10. The Math.random() method generates a random real numbers between 0 and 1. It returns the output in double data type. It can be used as a building block to create random numbers in different ranges by scaling and shifting the result.
11. In a class and object program, if the constructor is not mentioned, then the compiler calls the default constructor which assigns all the data members of the class to their default values.
12. The difference between Constructor and Method is as follows:

    Constructor: Initialises instance variables of the objects and has no return type. Its name matches the class name and is called automatically upon object creation.

    Method: Defines functionality for objects, has a return type, and is called explicitly to perform operations.

    Eg:

```
class sum
{
  int a,b,c;
```

```
  sum(int i, int j)   // Constructor
  {
   a= i ; b=j ;
  }
  void display()   // method
  {
   c=a+b;
   System.out.println(c);
  }
}
```

13. A copy constructor is a special type of constructor in object-oriented programming that initialises a new object as a copy of an existing object. It is particularly useful for creating a new instance that is a duplicate of another instance, ensuring that all the data members are copied correctly.

    Eg:

```
class Person {
    String name;
    int age;

    // Parameterised constructor
    Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    // Copy constructor
    Person(Person other) {
        this.name = other.name;
        this.age = other.age;
    }
}
```

14. A constructor is a special class member in object-oriented programming that serves several critical purposes for initialising objects. Constructors are used to initialise the state of an object at the time of its creation. Constructors can enforce certain invariants or constraints that are necessary for the class to function correctly. Constructors can provide default values for object attributes if no specific values are provided. Constructors can allocate resources or perform setup tasks necessary for the object. this are some of the reasons a constructor is required.

15. No, a constructor does not return a value. The purpose of a constructor is to set up an object's initial state, rather than to perform operations or return data.

    Eg.

```
class MyClass {
    // Constructor
    MyClass() {
        // Initialisation code
    }
}
```

16. SHOW Method..
17. CAKES=10
18. NEGATIVE
19. C .

    N .

    E .

    I .

    C .

    S .

    SCIENC
20. (i) ?1? should be i < n - 1 (outer loop condition)

    (ii) ?2? should be j < n - i - 1 (inner loop condition)

    (iii) ?3? should be arr[j + 1] (element to compare against)

    (iv) ?4? should be arr[j] (element being swapped)

    (v) ?5? should be temp (new value for arr[j + 1])
21. (i) ?1? should be i < a.length - 1 (condition on outer loop)

    (ii) ?2? should be i + 1 (start index of inner loop)

    (iii) ?3? should be minpos (index of the new minimum element)

    (iv) ?4? should be a[j] (new minimum value)

    (v) ?5? should be t (new value for a[i])
22. 21
23. (i) ?1? should be b != 0 (the condition for the loop)

    (ii) ?2? should be a % b (the remainder)

    (iii) ?3? should be r (assign remainder temporarily to r)

    (iv) ?4? should be b (assign remainder to b)

    (v) ?5? should be a (return the GCD)

24. Hello5
Hello4
Hello3
Hello2
Hello1
0
1
2
3
4

25. 76

D. 1.

```
import java.util.Scanner;
class ConsChange
 {
    // Data Members
    String word;
    int len;
    String shiftedWord;
    String changedWord;
    // Default Constructor
    ConsChange() {
        word = "";
        len = 0;
        shiftedWord = "";
        changedWord = "";
    }
    // Method to read a word in lowercase
    void readword() {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a word in lowercase: ");
        word = sc.nextLine();
        len = word.length();
    }
   // Method to shift consonants to the beginning and vowels to the end
```

```
void shiftcons()
{
    String consonants ="";
    String vowels = "";

    for (int i=0;i<word.length();i++)
    {
        char ch=word.charAt(i);
        if(Character.isLetter(ch))
        {
            if("aeiou".indexOf(ch)!=-1)
                vowels+=ch;
            else
                consonants+=ch;

        }
    }
    shiftedWord = consonants+vowels;
}
// Method to change the case of consonants in the shifted word
to uppercase
void changeword()
{
    for (int i=0;i<shiftedWord.length();i++)
    {
        char ch=shiftedWord.charAt(i);
        if(Character.isLetter(ch) && "aeiou".indexOf(ch)==-1)
            changedWord+=Character.toUpperCase(ch);
        else
            changedWord+=ch;
    }
}
// Method to display the original word, shifted word, and changed
word
```

```
    void show()
    {
        System.out.println("Original word: " + word);
        System.out.println("Shifted word: " + shiftedWord);
        System.out.println("Changed word: " + changedWord);
    }
    // Main function
    public static void main(String[] args) {
        ConsChange cc = new ConsChange();
        cc.readword();
        cc.shiftcons();
        cc.changeword();
        cc.show();
    }
}
```

2.
```
import java.util.Scanner;
class Calculate
{
    // Data Member
    int n;
    // Method to input a number
    void input() {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        n = sc.nextInt();
    }
    // Method to check if the number is composite
    boolean composite() {
        if (n <= 1) return false;
        int count = 0;
        for (int i = 1; i <= n; i++) {
            if (n % i == 0) count++;
        }
        return count > 2;
```

```
}
// Method to check if the number is a magic number
boolean magic() {
    int sum = n;
    while (sum >= 10) {
        sum = sumDigits(sum);
    }
    return sum == 1;
}
// Helper method to sum the digits of a number
private int sumDigits(int number) {
    int sum = 0;
    while (number > 0) {
        sum += number % 10;
        number /= 10;
    }
    return sum;
}
// Method to check and print the results
void check() {
    boolean isComposite = composite();
    boolean isMagic = magic();
    if (isComposite && isMagic) {
        System.out.println("Composite Magic number");
    } else if (isComposite) {
        System.out.println("Composite number");
    } else if (isMagic) {
        System.out.println("Magic number");
    } else {
        System.out.println("Not a Composite Magic number");
    }
}
// Main function to test the class
public static void main(String[] args) {
```

```
            Calculate calc = new Calculate();
            calc.input();
            calc.check();
        }
    }
```

3.
```
    import java.util.*;
    class Lucky {
        // Data Member
        int n;
        // Method to input a number
        void input() {
            Scanner sc = new Scanner(System.in);
            System.out.print("Enter a number: ");
            n = sc.nextInt();
        }
        // Method to generate and print lucky numbers less than n
        void generate() {
            // Initialize the array with numbers from 1 to n-1
            int size = n - 1;
            int[] numbers = new int[size];
            for (int i = 0; i < size; i++) {
                numbers[i] = i + 1;
            }
            int k = 2; // Starting step to remove every k-th number
            while (k <= size) {
                // Remove every k-th number
                int count = 0;
                for (int i = 0; i < size; i++) {
                    if ((i + 1) % k != 0) {
                        numbers[count++] = numbers[i];
                    }
                }
                // Update size of the array
                size = count;
```

```
                k++;
                // Resize the array to the new size
                int[] temp = new int[size];
                for (int i = 0; i < size; i++) {
                    temp[i] = numbers[i];
                }
                numbers = temp;
            }
            // Print lucky numbers
            System.out.print("The Lucky numbers less than " + n + " are:
            ");
            for (int i = 0; i < size; i++) {
                System.out.print(numbers[i] + " ");
            }
            System.out.println();
        }
        // Main function to test the class
        public static void main(String[] args) {
            Lucky lucky = new Lucky();
            lucky.input();
            lucky.generate();
        }
    }
```

4.
```
    import java.util.*;
    class Kaprekar {
        // Data Member
        int n;
        // Constructor to initialize n to 0
        Kaprekar() {
            n = 0;
        }
        // Method to input a number
        void input() {
            Scanner scanner = new Scanner(System.in);
```

```
        System.out.print("Enter a number: ");
        n = scanner.nextInt();
    }
    // Method to check if n is a Kaprekar number
    boolean generate_kaprekar() {
        // Calculate the square of the number
        long square = (long) n * n;
        String squareStr = Long.toString(square);
        int len = squareStr.length();
        int lenRightPart = 1;
        // Split the square into two parts and check
        while (lenRightPart <= len) {
            String rightPartStr = squareStr.substring(len -
            lenRightPart);
            String leftPartStr = squareStr.substring(0, len -
            lenRightPart);
            // Convert to integers
            int rightPart = Integer.parseInt(rightPartStr);
            int leftPart = leftPartStr.isEmpty() ? 0 : Integer.
            parseInt(leftPartStr);
            if (leftPart + rightPart == n && rightPart != 0) {
                return true;
            }
            lenRightPart++;
        }
        return false;
    }
    // Method to display whether the number is a Kaprekar number
    void display() {
        if (generate_kaprekar()) {
            System.out.println(n + " is a Kaprekar number.");
        } else {
            System.out.println(n + " is not a Kaprekar number.");
        }
    }
```

```
    // Main function to test the class
    public static void main(String[] args) {
        Kaprekar kaprekar = new Kaprekar();
        kaprekar.input();
        kaprekar.display();
    }
}
```

5.
```
class Perfect {
    // Data Member
    int num;
    // Parameterised Constructor
    Perfect(int nn) {
        num = nn;
    }
   // Method to calculate sum of factors excluding the number itself
using recursion
    private int sum_of_factors(int i) {
        // Base case: if i is greater than or equal to num, return 0
        if (i >= num) {
            return 0;
        }
        // Check if i is a factor of num
        if (num % i == 0) {
            // Add i to the sum of factors
            return i + sum_of_factors(i + 1);
        } else {
            // Skip i if it's not a factor
            return sum_of_factors(i + 1);
        }
    }
    // Method to check if the number is perfect and display the result
    void check() {
        int sum = sum_of_factors(1);
        if (sum == num) {
```

```
            System.out.println(num + " is a perfect number.");
        } else {
            System.out.println(num + " is not a perfect number.");
        }
    }
    // Main function to test the class
    public static void main(String[] args) {
        Perfect perfectNumber = new Perfect(28); // Example number
        perfectNumber.check(); // Check if the number is perfect
    }
}
```

6.
```
import java.util.*;
class Point {
    // Data Members
    double x, y;
    // Default Constructor
    Point() {
        x = 0;
        y = 0;
    }
    // Method to read coordinates of a point
    void readpoint() {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the x-coordinate: ");
        x = sc.nextDouble();
        System.out.print("Enter the y-coordinate: ");
        y = sc.nextDouble();
    }
    // Method to calculate and return the midpoint of two points
    Point midpoint(Point A, Point B) {
        Point mid = new Point();
        mid.x = (A.x + B.x) / 2;
        mid.y = (A.y + B.y) / 2;
        return mid;
```

```
    }
    // Method to display the coordinates of the point
    void displaypoint() {
        System.out.println("Coordinates: (" + x + ", " + y + ")");
    }
    // Main function to test the class
    public static void main(String[] args) {
        Point p1 = new Point();
        Point p2 = new Point();
        Point midPoint = new Point();
        System.out.println("Enter coordinates for Point 1:");
        p1.readpoint();
        System.out.println("Enter coordinates for Point 2:");
        p2.readpoint();
        // Calculate midpoint of p1 and p2
        midPoint = midPoint.midpoint(p1, p2);
        System.out.println("Point 1:");
        p1.displaypoint();
        System.out.println("Point 2:");
        p2.displaypoint();
        System.out.println("Midpoint:");
        midPoint.displaypoint();
    }
}
```

7.
```
import java.util.*;

class Combine {
    // Data Members
    int com[];
    int size;
    // Parameterised Constructor
    Combine(int nn) {
        size = nn;
        com = new int[size];
```

```
}
// Method to accept array elements
void inputarray() {
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter " + size + " elements:");
    for (int i = 0; i < size; i++) {
        com[i] = sc.nextInt();
    }
}
// Method to sort the elements using selection sort
void sort() {
    int n = com.length;
    for (int i = 0; i < n - 1; i++) {
        int minIndex = i;
        for (int j = i + 1; j < n; j++) {
            if (com[j] < com[minIndex]) {
                minIndex = j;
            }
        }
       // Swap the found minimum element with the first element
        int temp = com[minIndex];
        com[minIndex] = com[i];
        com[i] = temp;
    }
}
// Method to combine two arrays and store the result in the
current object array
void mix(Combine A, Combine B) {
    size = A.size + B.size;
    com = new int[size];
    int index = 0;
    // Copy elements from A
    for (int i = 0; i < A.size; i++) {
        com[index++] = A.com[i];
    }
```

```
        // Copy elements from B
        for (int i = 0; i < B.size; i++) {
            com[index++] = B.com[i];
        }
    }
    // Method to display the array elements
    void display() {
        for (int i = 0; i < com.length; i++) {
            System.out.print(com[i] + " ");
        }
        System.out.println();
    }
    // Main method to test the class
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        // Creating objects for two arrays
        System.out.print("Enter size for first array: ");
        int sizeA = sc.nextInt();
        Combine A = new Combine(sizeA);
        System.out.println("Input elements for first array:");
        A.inputarray();
        System.out.print("Enter size for second array: ");
        int sizeB = sc.nextInt();
        Combine B = new Combine(sizeB);
        System.out.println("Input elements for second array:");
        B.inputarray();
        // Combining and sorting
        Combine combined = new Combine(sizeA + sizeB);
        combined.mix(A, B);
        combined.sort();
        // Display the result
        System.out.println("Combined and sorted array:");
        combined.display();
    }
}
```

8.
```
import java.util.*;
class Mixer {
    // Data Members
   int[] arr;
      int n;
    // Parameterised Constructor
    Mixer(int nn) {
        n = nn;
        arr = new int[n];
    }
    // Method to accept elements of the array in ascending order
    without duplicates
    void accept() {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter " + n + " elements in ascending
        order without duplicates:");
        for (int i = 0; i < n; i++)
        {
            arr[i] = sc.nextInt();
        }
    }
    // Method to merge the current object's array with the parameterised
    array elements
    Mixer mix(Mixer A) {
        int merged[] = new int[n + A.n];
        int i = 0, j = 0, k = 0;
        // Merge the arrays
        while (i < this.n && j < A.n) {
            if (this.arr[i] < A.arr[j]) {
                merged[k++] = this.arr[i++];
            } else if (this.arr[i] > A.arr[j]) {
                merged[k++] = A.arr[j++];
            } else {
                // If both are equal, include only one occurrence
```

```
                merged[k++] = this.arr[i++];
                j++;
            }
        }
        // Copy remaining elements
        while (i < this.n) {
            merged[k++] = this.arr[i++];
        }
        while (j < A.n) {
            merged[k++] = A.arr[j++];
        }
        // Create a new Mixer object to return the merged array
        Mixer result = new Mixer(this.n + A.n);
        result.arr = merged; // Array copy
        return result;
    }
    // Method to display the elements of the array
    void display() {
        for (int i=0;i<arr.length;i++) {
            System.out.print(arr[i] + " ");
        }
        System.out.println();
    }
    // Main method to test the class
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        // Create and populate the first array
        System.out.print("Enter size for the first array: ");
        int size1 = sc.nextInt();
        Mixer A = new Mixer(size1);
        System.out.println("Input elements for the first array:");
        A.accept();
        // Create and populate the second array
        System.out.print("Enter size for the second array: ");
```

```
            int size2 = sc.nextInt();
            Mixer B = new Mixer(size2);
            System.out.println("Input elements for the second array:");
            B.accept();
            // Merge the arrays and display the result
            Mixer C = A.mix(B);
            System.out.println("Merged array:");
            C.display();
        }
    }
```

9.
```
    import java.util.*;
    class Disarium {
        // Data Members
        int num, size;
        // Parameterised Constructor
        Disarium(int nn) {
            num = nn;
            size = 0;
        }
        // Method to count the total number of digits and assign it to size
        void countDigit() {
            size = String.valueOf(num).length();
        }
        // Recursive method to calculate the sum of digits raised to
        their respective positions
        int sumofDigits(int n, int p) {
            if (n == 0) {
                return 0;
            } else {
                int digit = n % 10;
                return (int) Math.pow(digit, p) + sumofDigits(n / 10,
                p - 1);
            }
        }
```

```
    // Method to check whether the number is a Disarium number and
    display the result
    void check() {
        countDigit(); // Calculate the number of digits
        int sum = sumofDigits(num, size); // Calculate the sum of
        digits raised to their positions
        if (sum == num) {
            System.out.println(num + " is a Disarium number.");
        } else {
            System.out.println(num + " is not a Disarium number.");
        }
    }
    // Main method to test the class
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        // Input a number
        System.out.print("Enter a number to check if it is a Disarium
        number: ");
        int number = sc.nextInt();
        // Create an object of Disarium class
        Disarium disarium = new Disarium(number);
        // Check if the number is a Disarium number
        disarium.check();
    }
}
```

10.
```
import java.util.Scanner;
class Shift {
    // Data Members
    int mat[][];
    int m, n;
    // Parameterised Constructor
    Shift(int mm, int nn) {
        m = mm;
        n = nn;
        mat = new int[m][n];
```

```
}
// Method to enter elements of the array
void input() {
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter the elements of the " + m + "x" +
    n + " matrix:");
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            mat[i][j] = sc.nextInt();
        }
    }
}
// Method to shift the matrix rows upwards in a cyclic manner
void cyclic(Shift p) {
    // Create a new matrix to store the result of the cyclic shift
    int result[][] = new int[m][n];
    // Perform the cyclic shift
    for (int i = 0; i < m; i++) {
        int r = (i + 1) % m; // Calculate new row index
        for (int j = 0; j < n; j++) {
            result[r][j] = p.mat[i][j];
        }
    }
   mat = result; // Copy the result matrix to the current matrix
}
// Method to display the matrix elements
void display() {
    System.out.println("Matrix:");
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            System.out.print(mat[i][j] + " ");
        }
        System.out.println();
    }
```

```
    }
    // Main method to test the class
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        // Define the size of the matrix
        System.out.print("Enter the number of rows (m): ");
        int m = sc.nextInt();
        System.out.print("Enter the number of columns (n): ");
        int n = sc.nextInt();
        // Create an object of the Shift class
        Shift original = new Shift(m, n);
        // Input elements into the matrix
        original.input();
        // Display the original matrix
        System.out.println("Original matrix:");
        original.display();
        // Create another object for the shifted matrix
        Shift shifted = new Shift(m, n);
        // Perform the cyclic shift
        shifted.cyclic(original);
        // Display the shifted matrix
        System.out.println("Matrix after cyclic shift:");
        shifted.display();
    }
}
```

11.
```
class DeciOct
 {
    // Data Members
    int n;  // Stores the decimal number
    String oct;  // Stores the octal equivalent number as a String
    // Default Constructor
    DeciOct() {
        n = 0;
        oct = "";
```

```
}
// Method to assign the decimal number
void getnum(int nn) {
    n = nn;
}
// Recursive method to calculate the octal equivalent
void deci_oct() {
    oct = "";  // Initialize octal representation
    deci_octRecursive(n);  // Start recursion
}
// Helper recursive method to compute the octal equivalent
private void deci_octRecursive(int num) {
    if (num == 0) {
        return;
    } else {
        deci_octRecursive(num / 8);
       oct += (num % 8); // Append the remainder to octal string
    }
}
// Method to display the decimal and octal values
void show() {
    if (n == 0) {
        oct = "0";
    } else {
        deci_oct();  // Compute octal equivalent
    }
    System.out.println("Decimal Number: " + n);
    System.out.println("Octal Equivalent: " + oct);
}
// Main method to test the class
public static void main(int n) {
    DeciOct obj = new DeciOct();
    // Test the functionality
    obj.getnum(n);  // Example input
```

```
            obj.show();  // Display the decimal and its octal equivalent
        }
    }
```

12.
```
class ArmNum {
    // Data Members
   int n; // Stores the number
    int l; // Stores the length of the number
    // Parameterised Constructor
    ArmNum(int nn) {
        n = nn;
        l = String.valueOf(nn).length();  // Calculate the length
        of the number
    }
    // Recursive method to calculate the sum of each digit raised
    to the power of the length of the number
    int sum_pow(int num, int power) {
        if (num == 0) {
            return 0;
        } else {
            int digit = num % 10;  // Extract the last digit
            return (int) Math.pow(digit, power) + sum_pow(num / 10,
            power);  // Recursive call
        }
    }
    // Method to check if the number is an Armstrong number
    void isArmstrong() {
        int sum = sum_pow(n, l);  // Calculate the sum of each digit
        raised to the power of l
        if (sum == n) {
            System.out.println(n + " is an Armstrong number.");
        } else {
           System.out.println(n + " is not an Armstrong number.");
        }
    }
```

```
    // Main method to test the class
    public static void main(String[] args) {
        ArmNum obj1 = new ArmNum(371);  // Example input
        obj1.isArmstrong();  // Check and display whether the number
        is an Armstrong number
        ArmNum obj2 = new ArmNum(1634);  // Example input
        obj2.isArmstrong();  // Check and display whether the number
        is an Armstrong number
        ArmNum obj3 = new ArmNum(54748);  // Example input
        obj3.isArmstrong();  // Check and display whether the number
        is an Armstrong number
        ArmNum obj4 = new ArmNum(123);  // Example input
        obj4.isArmstrong();  // Check and display whether the number
        is an Armstrong number
    }
}
```

13.
```
class isbn
 {
    // Data Members
    int n;  // Stores the ISBN number
    // Parameterised Constructor
    isbn(int nn) {
        n = nn;
    }
    // Method to check if the number is a valid ISBN
    void is_isbn() {
        // Convert the integer to a string to process each digit
        String numStr = String.format("%010d", n);  // Ensure it's
        //10 digits long, padding with zeros if necessary
        // Check if the number is actually 10 digits long
        if (numStr.length() != 10) {
            System.out.println("The number is not a valid 10-digit
            ISBN.");
            return;
        }
```

```
        int sum = 0;
        // Calculate the weighted sum for ISBN validation
        for (int i = 0; i < 10; i++) {
            int digit = Character.getNumericValue(numStr.charAt(i));
            // Get the digit at position i
            sum += digit * (10 - i);  // Calculate the weighted sum
        }
        // Check if the sum is divisible by 11
        if (sum % 11 == 0) {
            System.out.println(n + " is a valid ISBN number.");
        } else {
            System.out.println(n + " is not a valid ISBN number.");
        }
    }
    // Main method to test the class
    public static void main(String[] args) {
        isbn book1 = new isbn(1259060977);  // Example valid ISBN
        book1.is_isbn();  // Check and display whether the number
        is a valid ISBN
        isbn book2 = new isbn(1234567890); // Example invalid ISBN
        book2.is_isbn(); // Check and display whether the number is
        a valid ISBN
    }
}
```

14.
```
import java.util.*;
class Adder {
    // Data member to hold hours and minutes
    int a[] = new int[2]; // a[0] for hours, a[1] for minutes
    // Default constructor
    Adder()
    {
        a[0] = 0; // Initialize hours to 0
        a[1] = 0; // Initialize minutes to 0
    }
    // Method to read time from user
```

```
void readtime() {
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter hours: ");
    a[0] = sc.nextInt();
    System.out.print("Enter minutes: ");
    a[1] = sc.nextInt();
}
// Method to add two Adder objects' times
void addtime(Adder X, Adder Y)
{
    // Adding hours and minutes
    a[0] = X.a[0] + Y.a[0];
    a[1] = X.a[1] + Y.a[1];
    // Converting minutes to hours if greater than or equal to 60
    if (a[1] >= 60)
    {
        a[0] += a[1] / 60; // Adding extra hours
        a[1] = a[1] % 60;  // Updating minutes
    }
}
// Method to display the time
void disptime()
{
    System.out.println("Total time: " + a[0] + " hours " + a[1]
    + " minutes");
}
// Main method to test the functionality
public static void main(String[] args)
{
    Adder time1 = new Adder();
    Adder time2 = new Adder();
    Adder sum = new Adder();
    System.out.println("Enter the first time:");
    time1.readtime();
    System.out.println("Enter the second time:");
```

```
            time2.readtime();
            sum.addtime(time1, time2);
            sum.disptime();
        }
    }
```

9. Arrays

Unsolved Questions

A. 1. a 2. b 3. c 4. c 5. c

B. 1. float 2. even 3. Lower Bound 4. 2 5. less

C. 1. (i) ?1?: a.length: Set m to a.length, which is the length of the array.

(ii) ?2?: 1: The outer loop starts from 1 because the first element is trivially "sorted."

(iii) ?3?: b: The inner loop checks if b is greater than or equal to 0 and whether the current value t is smaller than a[b].

(iv) ?4?: b–: Decrement b to move to the previous element.

(v) ?5?: a[b + 1]: Insert t into the correct position.

2. (i) ?1?: i < arr.length - 1: The outer loop runs from 0 to n - 1, where n is the length of the array. The i index tracks how many elements have been sorted.

(ii) ?2?: j < arr.length - i - 1: The inner loop runs from 0 to n - i - 1, since the last i elements are already sorted.

(iii) ?3?: arr [j + 1]: Check if arr[j] is greater than arr[j+1] to determine if a swap is needed.

(iv) ?4?: arr [j]: Assign the value arr[j+1] to arr[j] as part of the swap.

(v) ?5?: temp: Assign temp, which holds the value of arr[j], to arr[j+1].

3. 2

3

5

7

11

13

17

19

23

29

4. (i) JUNE

   JNUE

   (ii) SCROLL

   LCROLS

   LLROCS

   LLORCS

   (iii) function fun1() swaps elements in the array s[] from the outermost to the innermost positions and prints the array at each recursive step.

5. (i) ?1?: a.length - 1

   (ii) ?2?: i + 1

   (iii) ?3?: minpos

   (iv) ?4?: a[j]

   (v) ?5?: t

D. 1.

```
import java.util.*;
class SimpleFrequency
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        // Input the number of elements
        System.out.print("Enter the number of elements (n): ");
        int n = sc.nextInt();
        int[] arr = new int[n];
        boolean[] visited = new boolean[n];
        // Input the elements into the array
        System.out.println("Enter the elements of the array:");
        for (int i = 0; i < n; i++)
        {
            arr[i] = sc.nextInt();
            visited[i] = false; // Initialize the visited array to
            track processed elements
        }
        System.out.println("Frequency of each number:");
        for (int i = 0; i < n; i++)
        {
```

```
            if (visited[i])
            {
                continue; // Skip already processed elements
            }
            int count = 1; // Start with 1 occurrence of the current
            element
            for (int j = i + 1; j < n; j++)
            {
                if (arr[i] == arr[j])
                {
                    count++;
                    visited[j] = true;  // Mark the duplicate element
                    as processed
                }
            }
            // Print the element and its frequency
            System.out.println(arr[i] + " occurs " + count + "
            time(s)");
        }
    }
}
```

2.
```
import java.util.*;
class ArraySum
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        int[] arr = new int[20];  // Declare an array to store 20
        numbers
        int sum = 0;  // Variable to store the sum
        // Input 20 numbers into the array
        System.out.println("Enter 20 numbers:");
        for (int i = 0; i < 20; i++)
        {
```

```
            arr[i] = sc.nextInt();  // Store each input number in
            the array
            sum += arr[i];  // Add each number to the sum
        }
        // Print the sum of all numbers in the array
        System.out.println("The sum of all numbers in the array is:
        " + sum);
    }
}
```

3.
```
import java.util.*;
class ArrangeArray1 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int i, j, temp;
        // Input the number of elements (n)
        System.out.print("Enter the number of elements (n > 0): ");
        int n = sc.nextInt();
        // Create an array to store the integers
        int[] arr = new int[n];
        // Input the array elements
        System.out.println("Enter " + n + " elements:");
        for (i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }
        // Sort the array in ascending order
        for (i = 0; i < n - 1; i++) {
            for (j = 0; j < n - 1 - i; j++) {  // Corrected range
            for sorting
                if (arr[j] > arr[j + 1]) {
                    temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            }
```

```
        }
        // Rearrange the array as per the required pattern
        int mid = n / 2; // Find the center of the array
        int[] result = new int[n]; // Array to store the rearranged
        result
        // Place the smallest element in the center
        result[mid] = arr[0];
       // Start filling elements to the right and left of the center
        int left = mid - 1;  // Start placing to the left of the
        center
        int right = mid + 1; // Start placing to the right of the
        center
        boolean placeRight = true; // Toggle to alternate between
        right and left
        for (i = 1; i < n; i++) {
            if (placeRight) {
                result[right] = arr[i];
                right++;  // Move to the next position on the right
            } else {
                result[left] = arr[i];
                left--;  // Move to the next position on the left
            }
            placeRight = !placeRight; // Toggle between placing on
            the right and left
        }
        // Output the rearranged array
        System.out.println("Rearranged array:");
        for (i = 0; i < n; i++) {
            System.out.print(result[i] + " ");
        }
        System.out.println();
        sc.close();
    }
}
```

4.
```
import java.util.Scanner;
public class PackingCartons {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        // Accept the number of boxes
        System.out.print("Enter the number of boxes to be packed
        (N): ");
        int N = sc.nextInt();
        // Carton sizes
        int[] cartonSizes = {48, 24, 12, 6};
        int totalCartons = 0;
        int remainingBoxes = N;
        System.out.println("Breakup of cartons used:");
        // Calculate the number of cartons for each size
        for (int size=0;size<4;size++)
        {
            int cartonCount = remainingBoxes / cartonSizes[size];
            if (cartonCount > 0) {
                System.out.println(cartonSizes[size] + " * " + cartonCount
                + " = " + (cartonSizes[size] * cartonCount));
              remainingBoxes = remainingBoxes % cartonSizes[size];
                totalCartons += cartonCount;
            }
        }
        // If there are any remaining boxes less than 6, add an extra
        carton of size 6
        if (remainingBoxes ==6) {
            System.out.println("6 * 1 = 6");
            remainingBoxes = 0;
            totalCartons++;
        }
        // Display the remaining boxes (if any), total number of
        boxes, and total number of cartons
        System.out.println("Remaining boxes = " + remainingBoxes);
```

```
            System.out.println("Total number of boxes = " + N);
            System.out.println("Total number of cartons = " + totalCartons);
        }
    }
```

5.
```
import java.util.*;
class SquareMatrixPattern {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        // Accept the size of the matrix
        System.out.print("Enter Size: ");
        int n = sc.nextInt();
        // Check if n exceeds the maximum allowed size
        if (n > 10) {
            System.out.println("Size exceeds the maximum allowed
            limit of 10.");
            return;
        }
        // Accept the three characters
        System.out.print("First Character: ");
        char char1 = sc.next().charAt(0);
        System.out.print("Second Character: ");
        char char2 = sc.next().charAt(0);
        System.out.print("Third Character: ");
        char char3 = sc.next().charAt(0);
        // Create the matrix and fill it according to the pattern
        char[][] matrix = new char[n][n];
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                if (i == j || i + j == n - 1) {
                    matrix[i][j] = char3;  // Diagonals
                } else if (i < j && i + j < n - 1 || i > j && i + j
                > n - 1) {
                   matrix[i][j] = char1;  // Upper and lower halves
                } else {
```

```
                    matrix[i][j] = char2;  // Middle sides
                }
            }
        }
        // Display the matrix
        System.out.println("Output:");
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                System.out.print(matrix[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

6.
```
import java.util.*;
class EqMat {
    // Data members
    private int[][] a;
    private int m, n;
    // Constructor to initialize dimensions and array
    EqMat(int mm, int nn) {
        m = mm;
        n = nn;
        a = new int[m][n];
    }
    // Function to read the matrix elements
    void readarray() {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the elements of the matrix:");
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                a[i][j] = sc.nextInt();
            }
        }
```

```
}
// Function to check if two matrices are equal
int check(EqMat P, EqMat Q) {
    for (int i = 0; i < P.m; i++) {
        for (int j = 0; j < P.n; j++) {
            if (P.a[i][j] != Q.a[i][j]) {
                return 0; // Matrices are not equal
            }
        }
    }
    return 1; // Matrices are equal
}
// Function to print the matrix
void print() {
    System.out.println("Matrix:");
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            System.out.print(a[i][j] + " ");
        }
        System.out.println();
    }
}
// Main method to test the functionality
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    // Input matrix dimensions
    System.out.print("Enter the number of rows: ");
    int rows = sc.nextInt();
    System.out.print("Enter the number of columns: ");
    int cols = sc.nextInt();
    // Create two matrix objects
    EqMat mat1 = new EqMat(rows, cols);
    EqMat mat2 = new EqMat(rows, cols);
    // Input matrix elements
```

```
            System.out.println("Enter the elements for Matrix A:");
            mat1.readarray();
            System.out.println("Enter the elements for Matrix B:");
            mat2.readarray();
            // Display both matrices
            System.out.println("Matrix A:");
            mat1.print();
            System.out.println("Matrix B:");
            mat2.print();
            // Check if the matrices are equal
            if (mat1.check(mat1, mat2) == 1) {
                System.out.println("The matrices are equal.");
            } else {
                System.out.println("The matrices are not equal.");
            }
        }
    }
```

7.
```
import java.util.Scanner;
class Pseudoarithmetic {
    // Data members
    int n;          // Size of the sequence
    int[] a;        // Array to store the sequence
    int ans;        // Sum of the sequence
    boolean flag; // Status to check if it is a pseudo-arithmetic
    sequence
    // Default constructor
    public Pseudoarithmetic() {
        n = 0;
        a = null;
        ans = 0;
        flag = false;
    }
    // Accept method to assign values and read the sequence
    void accept(int nn) {
```

```
        n = nn;
        a = new int[n];
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter " + n + " elements for the
        sequence:");
        for (int i = 0; i < n; i++) {
            a[i] = sc.nextInt();
        }
    }
// Method to check if the sequence is a pseudo-arithmetic sequence
    boolean check() {
        // Check for odd or even number of elements
        int pairs = n / 2; // Number of pairs
        int middle = 0;
        // Calculate the expected sum
        if (n % 2 == 1) { // Odd length, handle middle element
            middle = a[n / 2] * 2;
        }
        // Check sums of corresponding pairs
        int expectedSum = a[0] + a[n - 1]; // First and last pair sum
        for (int i = 0; i < pairs; i++) {
            if (a[i] + a[n - i - 1] != expectedSum) {
                return false; // If any pair sum doesn't match the
                first pair
            }
        }
        ans = expectedSum * pairs; // Total sum
        if (n % 2 == 1) {
            ans += middle; // Add middle element for odd-length
            sequence
        }
        flag = true; // Sequence is pseudo-arithmetic
        return true;
    }
```

```
    // Method to print the result
    void printResult() {
        if (flag) {
            System.out.println("The sequence is a pseudo-arithmetic
            sequence.");
            System.out.println("Sum of the sequence: " + ans);
        } else {
            System.out.println("The sequence is not a pseudo-arithmetic
            sequence.");
        }
    }
    // Main method to test the class
    public static void main(String[] args) {
        Pseudoarithmetic ps = new Pseudoarithmetic();
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the size of the sequence: ");
        int size = sc.nextInt();
        // Accept the sequence
        ps.accept(size);
        // Check and print the result
        if (ps.check()) {
            ps.printResult();
        } else {
            ps.printResult();
        }
    }
}
```

8.
```
class Adder {
    int[] a = new int[2]; // Array to hold hours (a[0]) and minutes (a[1])
    // Method to read the time in hours and minutes
    void readtime(int hours, int minutes) {
        a[0] = hours;
        a[1] = minutes;
    }
```

```
    // Method to add two Adder objects and store the result in the
    current object
    void addtime(Adder time1, Adder time2) {
        this.a[0] = time1.a[0] + time2.a[0];  // Add hours
        this.a[1] = time1.a[1] + time2.a[1];  // Add minutes
        // Adjust if minutes exceed 60
        if (this.a[1] >= 60) {
            this.a[0] += this.a[1] / 60;  // Add extra hours
            this.a[1] = this.a[1] % 60;   // Remaining minutes
        }
    }
    // Method to display the time
    void displaytime() {
        System.out.println("Time: " + a[0] + " hours " + a[1] + "
        minutes");
    }
    public static void main(String[] args)
    {
        // Create objects for time1 and time2
        Adder time1 = new Adder();
        Adder time2 = new Adder();
        Adder resultTime = new Adder();
        // Read two time values
        time1.readtime(6, 35);
        time2.readtime(7, 45);
        // Add the two times and store in resultTime
        resultTime.addtime(time1, time2);
        // Display the resulting time
        resultTime.displaytime();
    }
}
```

9.
```
import java.util.*;
class Disarium
{
```

```
int[] arr = new int[20]; // Array of size 20 to store numbers
// Method to accept the array elements
void acceptArray()
{
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter 20 numbers:");
    for (int i = 0; i < 20; i++)
    {
        arr[i] = sc.nextInt();
    }
}
// Method to check if a number is a Disarium number
boolean isDisarium(int num)
{
    int sum = 0, temp = num, count ;
    count = Integer.toString(num).length(); // calculating the
    number of digits
    // Calculate the sum of digits raised to the power of their
    positions
    while (temp > 0)
    {
        int digit = temp % 10;
        sum += Math.pow(digit, count);
        temp /= 10;
        count--;
    }
    return sum == num;
}
// Method to print and sum all Disarium numbers in the array
void printDisariumNumbers() {
    int sumOfDisariumNumbers = 0;
    System.out.println("Disarium numbers in the array:");
    for (int i = 0; i < 20; i++) {
        if (isDisarium(arr[i])) {
```

```
                System.out.println(arr[i]);
                sumOfDisariumNumbers += arr[i]; // Sum of Disarium
                numbers
            }
        }
        System.out.println("Sum of Disarium numbers: " +
        sumOfDisariumNumbers);
    }
    public static void main(String[] args) {
        // Create an object of Disarium class
        Disarium disariumChecker = new Disarium();
        // Accept array elements
        disariumChecker.acceptArray();
        // Print Disarium numbers and their sum
        disariumChecker.printDisariumNumbers();
    }
}
```

10.
```
import java.util.Scanner;
public class SpiralMatrix {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        // Input for the size of the matrix (n must be odd)
        System.out.print("Enter the size of the square matrix (odd
        number): ");
        int n = sc.nextInt();
        if (n % 2 == 0) {
            System.out.println("Please enter an odd number.");
            return;
        }
        int[][] matrix = new int[n][n]; // Create an n x n matrix
        int num = 1; // Start with the first number
        int top = 0, bottom = n - 1; // Define boundaries
        int left = 0, right = n - 1;
        // Fill the matrix in a spiral order
```

```
        while (num <= n * n) {
            // Fill the top row (left to right)
            for (int i = left; i <= right && num <= n * n; i++) {
                matrix[top][i] = num++;
            }
            top++;
            // Fill the right column (top to bottom)
            for (int i = top; i <= bottom && num <= n * n; i++) {
                matrix[i][right] = num++;
            }
            right--;
            // Fill the bottom row (right to left)
            for (int i = right; i >= left && num <= n * n; i--) {
                matrix[bottom][i] = num++;
            }
            bottom--;
            // Fill the left column (bottom to top)
            for (int i = bottom; i >= top && num <= n * n; i--) {
                matrix[i][left] = num++;
            }
            left++;
        }
        // Display the spiral-filled matrix
        System.out.println("Spiral Matrix:");
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                System.out.printf("%4d", matrix[i][j]);
            }
            System.out.println();
        }
    }
}
```

11.
```
import java.util.*;
class Admission
```

```
{
    int[] Adno = new int[100];  // Array to store admission numbers
    // Constructor to initialize the array elements (optional step
    as Java does it by default)
    Admission()
    {
        for (int i = 0; i < 100; i++)
        {
            Adno[i] = 0;
        }
    }
    // Method to fill the array in ascending order
    void fillArray()
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter 100 admission numbers in ascending
        order:");
        for (int i = 0; i < 100; i++)
        {
            Adno[i] = sc.nextInt();
        }
    }
    // Recursive binary search method
    int binSearch(int l, int u, int v)
    {
        if (l > u)
        {
            return -1;  // Element not found
        }
        int mid = (l + u) / 2;
        if (Adno[mid] == v)
        {
            return 1;  // Element found
        } else if (Adno[mid] > v) {
```

```
            return binSearch(l, mid - 1, v);  // Search in the left
            half
        } else {
            return binSearch(mid + 1, u, v);  // Search in the right
            half
        }
    }
    public static void main(String[] args) {
        // Create an object of Admission class
        Admission admissionList = new Admission();
        // Call the method to fill the array with admission numbers
        admissionList.fillArray();
        // Input the admission number to be searched
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the admission number to search: ");
        int searchNumber = sc.nextInt();
        // Perform binary search
        int result = admissionList.binSearch(0, 99, searchNumber);
        // Display the result
        if (result == 1) {
            System.out.println("Admission number " + searchNumber +
            " found.");
        } else {
            System.out.println("Admission number " + searchNumber +
            " not found.");
        }
    }
}
```

12.
```
import java.util.*;
public class MatrixInterchange {
    // Method to input matrix elements
    void inputMatrix(int[][] matrix, int n) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter elements of the matrix:");
        for (int i = 0; i < n; i++) {
```

```
            for (int j = 0; j < n; j++) {
                matrix[i][j] = sc.nextInt();
            }
        }
    }
    // Method to display the matrix
    void displayMatrix(int[][] matrix, int n) {
        System.out.println("Matrix:");
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                System.out.print(matrix[i][j] + "\t");
            }
            System.out.println();
        }
    }
    // Method to interchange two rows
    void interchangeRows(int[][] matrix, int row1, int row2, int n) {
        for (int j = 0; j < n; j++) {
            int temp = matrix[row1][j];
            matrix[row1][j] = matrix[row2][j];
            matrix[row2][j] = temp;
        }
    }
    // Method to interchange two columns
    void interchangeColumns(int[][] matrix, int col1, int col2, int n)
    {
        for (int i = 0; i < n; i++) {
            int temp = matrix[i][col1];
            matrix[i][col1] = matrix[i][col2];
            matrix[i][col2] = temp;
        }
    }
    public static void main(String[] args)
    {
```

```
Scanner sc = new Scanner(System.in);
MatrixInterchange MI = new MatrixInterchange();
// Input the size of the matrix (n x n)
System.out.print("Enter the size of the square matrix: ");
int n = sc.nextInt();
int[][] matrix = new int[n][n];
// Input the matrix elements
MI.inputMatrix(matrix, n);
// Display the original matrix
System.out.println("Original Matrix:");
MI.displayMatrix(matrix, n);
// Input the rows to be interchanged
System.out.print("Enter the first row to interchange (0 to
" + (n-1) + "): ");
int row1 = sc.nextInt();
System.out.print("Enter the second row to interchange (0 to
" + (n-1) + "): ");
int row2 = sc.nextInt();
// Interchange the rows
MI.interchangeRows(matrix, row1, row2, n);
// Display the matrix after row interchange
System.out.println("Matrix after interchanging rows " + row1
+ " and " + row2 + ":");
MI.displayMatrix(matrix, n);
// Input the columns to be interchanged
System.out.print("Enter the first column to interchange (0
to " + (n-1) + "): ");
int col1 = sc.nextInt();
System.out.print("Enter the second column to interchange (0
to " + (n-1) + "): ");
int col2 = sc.nextInt();
// Interchange the columns
MI.interchangeColumns(matrix, col1, col2, n);
// Display the matrix after column interchange
System.out.println("Matrix after interchanging columns " +
```

```
            col1 + " and " + col2 + ":");
            MI.displayMatrix(matrix, n);
        }
    }
```

13.
```
import java.util.*;
class Matrix
{
    int[][] arr; // stores the matrix elements
    int m, n;    // stores the number of rows (m) and columns (n)
    // Constructor to initialize the size of the matrix
    Matrix(int mm, int nn) {
        m = mm;
        n = nn;
        arr = new int[m][n]; // Initializing the 2D array with size
        m x n
    }
    // Method to fill the matrix with user input
    void fillarray() {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the elements of the matrix (" + m
        + "x" + n + "): ");
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                arr[i][j] = sc.nextInt();
            }
        }
    }
    // Method to subtract the current matrix from the parameter
    matrix A
    Matrix SubMat(Matrix A) {
        Matrix result = new Matrix(m, n);  // Resultant matrix with
        same dimensions
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
```

```
                result.arr[i][j] = this.arr[i][j] - A.arr[i][j];  //
                Subtract corresponding elements
            }
        }
        return result;  // Return the resultant matrix
    }
    // Method to display the matrix
    void display() {
        System.out.println("Matrix (" + m + "x" + n + "):");
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                System.out.print(arr[i][j] + "\t");
            }
            System.out.println();
        }
    }
    // Main method to create objects and perform matrix subtraction
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        // Input the number of rows and columns for both matrices
        System.out.print("Enter the number of rows: ");
        int rows = sc.nextInt();
        System.out.print("Enter the number of columns: ");
        int cols = sc.nextInt();
        // Create two matrix objects
        Matrix mat1 = new Matrix(rows, cols);
        Matrix mat2 = new Matrix(rows, cols);
        // Fill both matrices
        System.out.println("Fill Matrix 1:");
        mat1.fillarray();
        System.out.println("Fill Matrix 2:");
        mat2.fillarray();
        // Perform matrix subtraction
        Matrix result = mat1.SubMat(mat2);
        // Display the matrices
```

```
            System.out.println("Matrix 1:");
            mat1.display();
            System.out.println("Matrix 2:");
            mat2.display();
            System.out.println("Resulting Matrix (Matrix 1 - Matrix
            2):");
            result.display();
        }
    }
```

14.
```
import java.util.*;
class AlphabetMatrix
 {

    // Method to check if a character is a vowel
    boolean isVowel(char ch) {
        ch = Character.toLowerCase(ch); // Convert to lowercase for
        uniform comparison
        return (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' ||
        ch == 'u');
    }
    // Method to input alphabets into the matrix
    void fillArray(char[][] matrix, int m, int n) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the alphabets in the matrix (" +
        m + "x" + n + "): ");
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                matrix[i][j] = sc.next().charAt(0); // Input a single
                character
            }
        }
}
    // Method to replace vowels with 'V' and consonants with 'C'
    void replaceVowelsAndConsonants(char[][] matrix, int m, int n)
    {
```

```
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            if (isVowel(matrix[i][j])) {
                matrix[i][j] = 'V';  // Replace vowel with 'V'
            }
            else
            {
                matrix[i][j] = 'C';  // Replace consonant with
                'C'
            }
        }
    }
}
// Method to display the matrix
void displayMatrix(char[][] matrix, int m, int n)
{
    System.out.println("Modified Matrix:");
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            System.out.print(matrix[i][j] + "\t");
        }
        System.out.println();
    }
}
public static void main(String[] args)
{
    Scanner sc = new Scanner(System.in);
    AlphabetMatrix AM = new AlphabetMatrix();
    // Input the size of the matrix (m x n)
    System.out.print("Enter the number of rows: ");
    int m = sc.nextInt();
    System.out.print("Enter the number of columns: ");
    int n = sc.nextInt();
    // Create a 2D array to store alphabets
```

```
        char[][] matrix = new char[m][n];
        // Fill the array with alphabets
        AM.fillArray(matrix, m, n);
        // Replace vowels with 'V' and consonants with 'C'
        AM.replaceVowelsAndConsonants(matrix, m, n);
        // Display the modified matrix
        AM.displayMatrix(matrix, m, n);
    }
}
```

15.
```
import java.util.Scanner;
public class SaddlePoint {
    // Method to find the saddle point in a matrix
    static void findSaddlePoint(int[][] matrix, int rows, int cols)
    {
        boolean saddlePointFound = false;
        // Loop through each row
        for (int i = 0; i < rows; i++) {
            // Find the smallest element in the current row
            int minRowElement = matrix[i][0];
            int colIndex = 0;  // Store the column index of the
            smallest element
            for (int j = 1; j < cols; j++) {
                if (matrix[i][j] < minRowElement) {
                    minRowElement = matrix[i][j];
                    colIndex = j;  // Update the column index of the
                    smallest element
                }
            }
            // Check if the minimum element in the row is the largest
            in its column
            boolean isSaddlePoint = true;
            for (int k = 0; k < rows; k++) {
                if (matrix[k][colIndex] > minRowElement) {
                    isSaddlePoint = false;  // If any element in the
                    column is larger, it's not a saddle point
```

```
                    break;
                }
            }
            // If the saddle point is found
            if (isSaddlePoint) {
                System.out.println("Saddle Point found: " + minRowElement
                + " at position (" + i + "," + colIndex + ")");
                saddlePointFound = true;
            }
        }
        if (!saddlePointFound) {
         System.out.println("No Saddle Point found in the matrix.");
        }
    }
    // Main method to input the matrix and find the saddle point
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        // Define the size of the matrix
        int rows = 3;
        int cols = 4;
        int[][] matrix = new int[rows][cols];
        // Input the elements of the matrix
        System.out.println("Enter the elements of the matrix (3x4):
        ");
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                matrix[i][j] = sc.nextInt();
            }
        }
        // Display the matrix
        System.out.println("The matrix is: ");
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                System.out.print(matrix[i][j] + "\t");
```

```
                }
                System.out.println();
            }
            // Find and print the saddle point
            findSaddlePoint(matrix, rows, cols);
        }
    }
```

16.
```
import java.util.*;
public class Sort {
    // Data members
    int[] arr = new int[50];  // Array to store 50 integers
    int item;  // The item to search for in the array
    // Method to input 50 integers with no duplicates
    void inpdata() {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter 50 unique integers:");
        for (int i = 0; i < arr.length; i++) {
            int num;
            boolean isDuplicate;
            do {
                isDuplicate = false;
                num = sc.nextInt();  // Input an integer
                // Check for duplicates
                for (int j = 0; j < i; j++) {
                    if (arr[j] == num) {
                        isDuplicate = true;
                        System.out.println("Duplicate number! Please
                        enter another number:");
                        break;
                    }
                }
            } while (isDuplicate);  // Keep asking until no duplicate
            is entered
            arr[i] = num;  // Store the unique number in the array
```

```
    }
}
// Method to sort the array using bubble sort
void bubsort() {
    int n = arr.length;
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            // Swap if the current element is greater than the
            next element
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
    // Display the sorted array
    System.out.println("Sorted array using bubble sort:");
    for (int i = 0; i < n; i++) {
        System.out.print(arr[i] + " ");
    }
    System.out.println();
}
// Method to input an item to search for using binary search
void binsearch() {
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter the number to search for: ");
    item = sc.nextInt();
    // Perform binary search
    technique();
}
// Method to search the item using binary search technique and
display the result
void technique() {
```

```
        int low = 0;
        int high = arr.length - 1;
        boolean found = false;
        while (low <= high) {
            int mid = (low + high) / 2;
            if (arr[mid] == item) {
                System.out.println("Item " + item + " found at
                position: " + (mid + 1));
                found = true;
                break;
            } else if (arr[mid] < item) {
                low = mid + 1;
            } else {
                high = mid - 1;
            }
        }
        if (!found) {
            System.out.println("Item " + item + " not found in the
            array.");
        }
    }
    // Main method (for demonstration, though not needed as per the
    problem)
    public static void main(String[] args) {
        Sort obj = new Sort();
        // Input 50 unique integers
        obj.inpdata();
        // Sort the array using bubble sort
        obj.bubsort();
        // Search for a specific number using binary search
        obj.binsearch();
    }
}
```

17.
```
class indexedarray
{
    // Data member
    int[] arr = new int[100];  // Array to store 100 employee codes
    // Constructor to initialize the employee codes to 0
    indexedarray()
    {
        for (int i = 0; i < arr.length; i++) {
            arr[i] = 0;
        }
    }
    // Method to sort the array using selection sort technique
    void sortarr() {
        int n = arr.length;
        // Selection sort
        for (int i = 0; i < n - 1; i++) {
            // Find the minimum element in unsorted part of array
            int minIndex = i;
            for (int j = i + 1; j < n; j++) {
                if (arr[j] < arr[minIndex]) {
                    minIndex = j;
                }
            }
           // Swap the found minimum element with the first element
            int temp = arr[minIndex];
            arr[minIndex] = arr[i];
            arr[i] = temp;
        }
        // Display sorted array (Optional, can be removed)
        System.out.println("Sorted employee codes using selection
        sort:");
        for (int i = 0; i < n; i++) {
            System.out.print(arr[i] + " ");
        }
```

```
        System.out.println();
    }
    // Method to search for an employee code using binary search
    int binarysearch(int[] arr, int item) {
        int low = 0;
        int high = arr.length - 1;
        // Binary search implementation
        while (low <= high) {
            int mid = (low + high) / 2;
            if (arr[mid] == item) {
                return 1;  // Employee code found
            } else if (arr[mid] < item) {
                low = mid + 1;  // Search in the right half
            } else {
                high = mid - 1;  // Search in the left half
            }
        }
        return 0;  // Employee code not found
    }
    // Other member functions like fillarray() assumed to be written
    for you
}
```

18.
```
class change {
    // Data Members
    int[] a = new int[20]; // Array to store hexadecimal digits (at
    most 20 digits)
    int n;                  // Integer to be converted to base 16
    // Constructor to initialise instance variables
    change() {
        n = 0; // Initial value of 'n' is set to 0
    }
    // Method to accept an integer to be converted to base 16
    void input(int num) {
        n = num; // Assign the input number to 'n'
```

```
}
// Method to convert decimal integer to hexadecimal form
void hexadeci(int num) {
    int temp = num;
    int index = 0;
    System.out.print("The hexadecimal equivalent of " + num + "
    is: ");
    // Converting decimal to hexadecimal
    while (temp > 0) {
        int remainder = temp % 16; // Find the remainder when
        divided by 16
        // Store the remainder (hex digit) in the array
        a[index] = remainder;
        index++;
        temp = temp / 16; // Divide the number by 16
    }
    // Display hexadecimal in reverse order (most significant
    digit first)
    for (int i = index - 1; i >= 0; i--) {
        if (a[i] < 10) {
            // For digits 0-9
            System.out.print(a[i]);
        } else {
            // For A-F (10-15)
            System.out.print((char) (a[i] - 10 + 'A')); // Convert
            10-15 to 'A'-'F'
        }
    }
    System.out.println();
}
// Method to convert a hexadecimal number (stored in array) back
to decimal form
void decihexa(String hexNum) {
    int length = hexNum.length();
    int decimalValue = 0;
```

```
        // Converting hexadecimal back to decimal
        for (int i = 0; i < length; i++) {
            char hexChar = hexNum.charAt(i);
            int hexDigit;
            // Convert the hex character to the corresponding integer
            value
            if (hexChar >= '0' && hexChar <= '9') {
                hexDigit = hexChar - '0'; // Convert '0'-'9' to 0-9
            } else {
                hexDigit = hexChar - 'A' + 10; // Convert 'A'-'F'
                to 10-15
            }
            // Calculate the decimal value by multiplying with the
            appropriate power of 16
            decimalValue = decimalValue * 16 + hexDigit;
        }
        // Display the decimal equivalent
        System.out.println("The decimal equivalent of hexadecimal "
        + hexNum + " is: " + decimalValue);
    }
    // Assume the main function and other helper functions are
    provided or written
}
```

19.
```
import java.util.*;
class Shift
 {
    // Data Members
    int[][] mat; // 2D array to store matrix elements
    int m, n;    // Dimensions of the matrix (rows and columns)
    // Constructor to initialize the matrix and its dimensions
    Shift(int mm, int nn) {
        m = mm;
        n = nn;
      mat = new int[m][n]; // Initialize 2D array based on dimensions
    }
```

```
// Method to input elements of the matrix
void input() {
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter the elements of the matrix (" + m
    + " x " + n + "): ");
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            mat[i][j] = sc.nextInt(); // Read elements row-wise
        }
    }
}
// Method to perform cyclic shift of rows upwards
void cyclic(Shift P) {
    // Copy the rows from object P's matrix to this matrix with
    a cyclic shift
    for (int i = 0; i < m - 1; i++) {
        mat[i] = P.mat[i + 1]; // Shift the rows upwards
    }
    mat[m - 1] = P.mat[0]; // The first row of P becomes the
    last row of the current matrix
}
// Method to display the matrix
void display() {
    System.out.println("Matrix after cyclic row shift: ");
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            System.out.print(mat[i][j] + "\t"); // Print each
            element
        }
        System.out.println(); // Newline after each row
    }
}
// Main function to demonstrate the functionality
public static void main(String[] args) {
```

```
        // Define the matrix size (m = 3, n = 3 as an example, can
        go up to 5x5)
        Shift matrix1 = new Shift(3, 3); // Create object matrix1
        with dimensions 3x3
        Shift matrix2 = new Shift(3, 3); // Create object matrix2
        to store the shifted matrix
        // Input the matrix elements for matrix1
        matrix1.input();
        // Perform the cyclic shift on matrix1 and store the result
        in matrix2
        matrix2.cyclic(matrix1);
        // Display the result of the shifted matrix
        matrix2.display();
    }
}
```

20.
```
import java.util.*;
class ArrayOperations
{
    public static void main(String[] args)
    {
        int arr[][] = new int[4][3];
        Scanner sc = new Scanner(System.in);
        // Input elements in the array
        System.out.println("Enter the elements of a 4x3 array:");
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 3; j++) {
                arr[i][j] = sc.nextInt();
            }
        }
        // a. Product of all even numbers in odd positions
        int productEvenOddPositions = 1;
        boolean evenFound = false;
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 3; j++) {
```

```
                int position = i * 3 + j + 1; // Calculate 1D position
                in row-major order
                if (position % 2 != 0 && arr[i][j] % 2 == 0) { //
                Odd position and even number
                    productEvenOddPositions *= arr[i][j];
                    evenFound = true;
                }
            }
        }
        if (evenFound) {
            System.out.println("Product of all even numbers in odd
            positions: " + productEvenOddPositions);
        } else {
            System.out.println("No even numbers found in odd
            positions.");
        }
        // b. Print all non-boundary elements
        System.out.println("Non-boundary elements:");
        for (int i = 1; i < 3; i++) { // Non-boundary rows (1 and
        2)
            for (int j = 1; j < 2; j++) { // Non-boundary columns
            (1)
                System.out.print(arr[i][j] + " ");
            }
        }
    }
}
```

21.
```
import java.util.*;
class MatrixOperations
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        // Input the size of the matrix
        int M, N;
```

```
do {
    System.out.print("Enter the number of rows (M) [3-19]:
    ");
    M = sc.nextInt();
    System.out.print("Enter the number of columns (N) [3-
    19]: ");
    N = sc.nextInt();
} while (M < 3 || M > 19 || N < 3 || N > 19); // Ensuring
valid M and N
int[][] A = new int[M][N];
// Input elements in the matrix
System.out.println("Enter the elements of the matrix (" + M
+ "x" + N + "):");
for (int i = 0; i < M; i++) {
    for (int j = 0; j < N; j++) {
        A[i][j] = sc.nextInt();
    }
}
// Display the input matrix
System.out.println("Original matrix:");
for (int i = 0; i < M; i++) {
    for (int j = 0; j < N; j++) {
        System.out.print(A[i][j] + " ");
    }
    System.out.println();
}
// Find the maximum and minimum values in the matrix along
with their positions
int max = A[0][0];
int min = A[0][0];
int maxRow = 0, maxCol = 0;
int minRow = 0, minCol = 0;
for (int i = 0; i < M; i++) {
    for (int j = 0; j < N; j++) {
        if (A[i][j] > max) {
```

```
                    max = A[i][j];
                    maxRow = i;
                    maxCol = j;
                }
                if (A[i][j] < min) {
                    min = A[i][j];
                    minRow = i;
                    minCol = j;
                }
            }
        }
        // Display the results
        System.out.println("Largest Number: " + max + " Row: " +
        maxRow + " Column: " + maxCol);
        System.out.println("Smallest Number: " + min + " Row: " +
        minRow + " Column: " + minCol);
    }
}
```

22.
```
import java.util.Scanner;
public class MatrixSort {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        // Input matrix dimensions
        System.out.print("Enter number of rows (M): ");
        int M = sc.nextInt();
        System.out.print("Enter number of columns (N): ");
        int N = sc.nextInt();
        // Check if M and N are within the valid range
        if (M <= 2 || N <= 2 || M >= 20 || N >= 20) {
            System.out.println("M and N must be greater than 2 and
            less than 20.");
            return;
        }
        int[][] A = new int[M][N];
```

```
int[] tempArray = new int[M * N]; // To store all elements
in a 1D array for sorting
// Input elements into the matrix
System.out.println("Enter the elements of the matrix:");
int index = 0;
for (int i = 0; i < M; i++) {
    for (int j = 0; j < N; j++) {
        A[i][j] = sc.nextInt();
        tempArray[index++] = A[i][j]; // Store in 1D array
        for sorting
    }
}
// Display the original matrix
System.out.println("Original matrix:");
displayMatrix(A, M, N);
// Sort the 1D array
for(int i=0;i<(M*N)-1;i++)
{
   for(int j=0;j<(M*N)-1-i;j++)
   {
       if(tempArray[j]>tempArray[j+1])
       {
           int t=tempArray[j];
           tempArray[j]=tempArray[j+1];
           tempArray[j+1]=t;
       }
   }
}
// Rearrange the sorted elements back into the matrix
index = 0;
for (int i = 0; i < M; i++) {
    for (int j = 0; j < N; j++) {
        A[i][j] = tempArray[index++];
    }
```

```
            }
            // Display the rearranged matrix
            System.out.println("Rearranged matrix:");
            displayMatrix(A, M, N);
        }
        // Function to display the matrix
        public static void displayMatrix(int[][] matrix, int M, int N)
        {
            for (int i = 0; i < M; i++) {
                for (int j = 0; j < N; j++) {
                    System.out.print(matrix[i][j] + " ");
                }
                System.out.println();
            }
        }
    }
```

23.
```
    import java.util.*;
    public class MergeSortedArrays {
        public static void main(String[] args) {
            // Given sorted arrays A and B
            int[] A = {1, 5, 6, 7, 8, 10};
            int[] B = {2, 4, 9};
            // Call the merge function
            mergeArrays(A, B);
            // Output the result
            System.out.println("Sorted Arrays:");
            System.out.print(" A : ");
            for(int i=0;i<A.length;i++)
            {
                System.out.print(A[i]+" ");
            }
            System.out.print("\n B : ");
            // Copy the remaining q elements into B
            for(int i=0;i<B.length;i++)
```

```
    {
        System.out.print(B[i]+" ");
    }
}
// Function to merge two sorted arrays while maintaining sorted
order
public static void mergeArrays(int[] A, int[] B) {
    int p = A.length;
    int q = B.length;
    int[] temp = new int[p + q];  // Create a temporary array
    to hold both A and B
    int i = 0, j = 0, k = 0;
    // Merge both arrays into temp while keeping them sorted
    while (i < p && j < q) {
        if (A[i] <= B[j]) {
            temp[k++] = A[i++];
        } else {
            temp[k++] = B[j++];
        }
    }
    // Copy remaining elements of A, if any
    while (i < p) {
        temp[k++] = A[i++];
    }
    // Copy remaining elements of B, if any
    while (j < q) {
        temp[k++] = B[j++];
    }
    // Copy the first p elements into A
    for(i=0;i<p;i++)
    {
        A[i]=temp[i];
    }
    // Copy the remaining q elements into B
```

```
            for(i=0;i<q;i++)
            {
                B[i]=temp[p+i];
            }
        }
    }
```

24.
```
import java.util.*;
public class MissingLetter {
    public static void main(String[] args) {
        Scanner sc= new Scanner(System.in);
        int i,n;
        System.out.print("Enter number of elements ");
        n=sc.nextInt();
        // Example array of characters
        char letters[] =new char[n];
        for(i=0;i<n;i++)
        {
            System.out.println("Enter a character : ");
            letters[i]=sc.next().charAt(0);
        }
        // Call the function to find the missing letter
        char missingLetter = findMissingLetter(letters);
        // Output the missing letter
        System.out.println("Missing letter in the array: " +
        missingLetter);
    }
    // Function to find the missing letter
    public static char findMissingLetter(char[] letters) {
        for (int i = 0; i < letters.length - 1; i++) {
            // Check the difference between consecutive letters
            if (letters[i + 1] - letters[i] != 1) {
                // Return the missing letter
                return (char) (letters[i] + 1);
            }
```

```
            }
            // If no missing letter is found (shouldn't happen with the
            assumption)
            return '\0';  // Null character as a fallback
        }
    }
```

D. c. Assertion is true and Reason is false.

10. Strings

Unsolved Questions

A. 1. d 2. c 3. a 4. a 5. b 6. a
7. c 8. b 9. c

B.
1. 32
2. string
3. 'Science'
4. 'Value : 0'
5. 'delete()' or 'deleteCharAt()'
6. 'nextElement()'
7. The 'StringTokenizer'
8. 'compareToIgnoreCase()'
9. 'Rama as good an dancang'
10. concatenate

C. 1.

```
import java.util.*;
class palindrome {
    // Data members
    String wd;     // String to be checked
    String revwd;  // String to keep the reverse word
    // Member method to input the word
    void input() {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the word: ");
        wd = sc.nextLine();
```

```
        revwd = "";  // Initialize revwd as an empty string
    }
    // Member method to reverse the word and store in revwd
    void reverse() {
    for (int i = wd.length() - 1; i >= 0; i--) {
        revwd += wd.charAt(i);  // Append each character from the
        end of wd to revwd
    }
}
    // Member method to check if the word is a palindrome and display
    the result
    void display() {
        reverse();  // Call the reverse method
        if (wd.equalsIgnoreCase(revwd)) {
            System.out.println(wd + " is a palindrome.");
        } else {
            System.out.println(wd + " is not a palindrome.");
        }
    }
    // Main method to run the program
    public static void main(String[] args) {
        palindrome p = new palindrome();  // Create an instance of
        the palindrome class
        p.input();   // Input the word
        p.display(); // Check and display the result
    }
}
```

2.
```
import java.util.*;
class AlternateStrings {
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        // Accept two strings from the user
        System.out.print("Enter the first string: ");
```

```
        String str1 = sc.nextLine();
        System.out.print("Enter the second string: ");
        String str2 = sc.nextLine();
        // Ensure both strings have the same length
        if (str1.length() != str2.length()) {
            System.out.println("Both strings must have the same
            length.");
            return; // Exit the program if lengths do not match
        }
        String newStr = "";  // To store the new string
        // Build the new string by alternating characters
        int length = str1.length();
        for (int i = 0; i < length; i++) {
            newStr += str1.charAt(i);
        // Take from str1 left to right
            newStr += str2.charAt(length - 1 - i);
        // Take from str2 right to left
        }
        // Display the resulting string
        System.out.println("New String: " + newStr);
    }
}
```

3.
```
import java.util.Scanner;
public class Exchange
{
    String sent,rev;
    int size;
    Exchange()
    {
        sent="";
        rev="";
        size=0;
    }
    void readsentence()
```

```
{
    Scanner sc=new Scanner(System.in);
    System.out.print("Enter a sentence");
    sent=sc.nextLine();
    size=sent.length();
}
void exfirstlast()
{
    char ch=' ';
    String wd="";
    for(int i=0;i< size;i++)
    {
        ch=sent.charAt(i);
        if(ch==' '||ch=='.')
        {
            if(wd.length()>1)
            {
                rev=rev  +  wd.charAt(wd.length()-1);/*last
                character*/
                rev=rev + wd.substring(1,wd.length()-1);/*middle
                characters*/
                rev=rev + wd.charAt(0); /*first character*/
                rev=rev+ch;/*space or full stop*/
            }
            else
            {
                rev=rev+wd+ch;
            }
            wd="";
        }
        else
        {
            wd=wd+ch;
        }
```

```
        }
    }
    void display()
    {
        System.out.println(rev);
    }
    public static void main(String[] args)
    {
        Exchange obj1=new Exchange();
        obj1.readsentence();
        obj1.exfirstlast();
        obj1.display();
    }
}
```

4.
```
import java.util.*;
class Rearrange
{
    String Txt;
    String Cxt;
    int Len;
    public Rearrange ()
    {
        Txt= "";
        Cxt = "" ;
        Len = 0;
    }
    public void readword ()
    {
        Scanner sc= new Scanner(System.in);
        System.out.println( "Enter the String");
        Txt = sc.nextLine().toUpperCase();
    }
    public void convert()
    {
```

```
        boolean check = false;
        char ch1;
        ch1 = Txt.charAt(0);
        if (ch1 == 'A' || ch1 == 'E' || ch1 =='I' || ch1 == 'O' ||
        ch1 == 'U')
        {
            Cxt = Txt + "Y";
        }
        else
        {
            for (int i = 0; i < Txt.length(); i++)
            {
                ch1 = Txt.charAt(i);
                if("AEIOU".indexOf(ch1)!=-1)
                {
                    check = true;
                    Cxt = Txt.substring(i)+Txt.substring(0,i)+"C";
                    break;
                }
            }
            if (check == false)
            {
                Cxt = Txt + "N";
            }
        }
    }
    public void display()
    {
        System.out.println("The original string is " + Txt);
        System.out.println(" The new String is "+ Cxt);
    }
    public static void main(String[] args)
    {
        Rearrange obj = new Rearrange();
```

```
            obj.readword( );
            obj.convert( );
            obj.display();
        }
    }
```

5.
```
import java.util.*;
class SortWord {
    // Data members
    String txt;  // To store the word
    int len;     // To store the length of the word
    // Constructor to initialize the data members
    SortWord() {
        txt = "";  // Initialize the word
        len = 0;   // Initialize the length to zero
    }
    // Method to accept the word in lowercase
    void readTxt() {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a word in lowercase: ");
        txt = sc.nextLine().toLowerCase();  // Read the word and
        convert it to lowercase
        len = txt.length();  // Calculate the length of the word
    }
    // Method to sort the word in alphabetical order using bubble
    sort
    void sortTxt()
    {
        int i,j;
        char ch;
        String ntxt="";
        for(i=97;i<=122;i++)
        {
            for (j = 0; j < len ; j++)
            {
```

```
                ch=txt.charAt(j);
                if((char)i==ch)
                {
                    ntxt=ntxt+ch;
                }
            }
        }
       // txt = ntxt;  // Convert the sorted array back to a string
        System.out.println("Sorted word: " + ntxt);  // Display the
        sorted word
    }
    // Method to change the case of vowels to uppercase in the word
    void changeTxt() {
        String ntxt="";
        for (int i = 0; i < len; i++) {
            char ch = txt.charAt(i);
            if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' ||
            ch == 'u') {
                ntxt=ntxt+(char)(ch-32);   // Convert vowels to
                uppercase
            } else {
                ntxt=ntxt+ch;  // Keep other characters unchanged
            }
        }
        txt = ntxt;  // Store the changed word
    }
    // Method to display the changed string
    void disp()
    {
        System.out.println("Changed word with vowels in uppercase:
        " + txt);
    }
}
```

6.
```
import java.util.*;
class Mystring {
    // Data members
    String str;  // To store the string
    int len;     // To store the length of the string
    // Constructor to initialize the data members
    Mystring() {
        str = "";  // Initialize the string as empty
        len = 0;   // Initialize the length to zero
    }
    // Method to read the string from input
    void readstring() {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a string: ");
        str = sc.nextLine();  // Read the input string
        len = str.length();   // Calculate the length of the string
    }
    // Method to return the Unicode for the character at the given
    index
    int code(int index) {
        if (index >= 0 && index < len) {
            return (int) str.charAt(index);  // Return the Unicode
            of the character at index
        } else {
            System.out.println("Index out of bounds.");
            return -1;  // Return -1 if index is out of bounds
        }
    }
    // Method to display the longest word in the string
    void word() {
        String longestWord = "";
        String currentWord = "";
        for (int i = 0; i < str.length(); i++) {
            char c = str.charAt(i);
```

```
                if (Character.isLetter(c)) {
                    currentWord += c;
                } else {
                    if (currentWord.length() > longestWord.length()) {
                        longestWord = currentWord;
                    }
                    currentWord = "";
                }
            }
            // Check if the last word is the longest
            if (currentWord.length() > longestWord.length()) {
                longestWord = currentWord;
            }
            System.out.println("The longest word is: " + longestWord);
        }
    }
```

7.
```
    import java.util.*;
    class Check {
        // Data members
        String st;    // To store the string
        int cap;      // Counter for capital letters
        int small;    // Counter for small letters
        int len;      // To store the length of the string
        // Parameterized constructor to initialize the string and other
        data members
        Check(String s) {
            st = s;                  // Assign the string to st
           cap = 0;                // Initialize capital letter counter
            small = 0;              // Initialize small letter counter
            len = st.length();       // Set the length of the string
        }
        // Recursive function to find total number of capital and small
        letters
        void recletters() {
```

```
        countLetters(0);  // Start recursive counting from the first
        character (index 0)
    }
    // Helper method for recursion (to count letters)
    private void countLetters(int index) {
        if (index == len) {
            // Base case: when the entire string has been checked
            return;
        } else {
            char ch = st.charAt(index);  // Get the character at the
            current index
            if (Character.isUpperCase(ch)) {
                cap++;  // Increment capital letter counter
            } else if (Character.isLowerCase(ch)) {
                small++;  // Increment small letter counter
            }
          countLetters(index + 1);  // Recur for the next character
        }
    }
    // Method to display the count of capital and small letters
    void display() {
        System.out.println("Number of capital letters: " + cap);
        System.out.println("Number of small letters: " + small);
    }
}
```

8.
```
import java.util.*;
class ConsecutiveLetters {
    // Method to find and print consecutive letters in the sentence
    public static void findConsecutiveLetters(String sentence)
    {
        sentence = sentence.toUpperCase(); // Convert sentence to
        uppercase for uniformity
        String result = "";  // To store the result
        for (int i = 0; i < sentence.length() - 1; i++)
```

```
            {
                char current = sentence.charAt(i);
                char next = sentence.charAt(i + 1);
                // Check if both characters are letters and consecutive
                if (Character.isLetter(current) && Character.isLetter(next)
                && (next - current == 1)) {
                    result += current + "" + next + ", ";   // Add
                    consecutive pair to the result
                }
            }
            // Remove trailing comma and space, if any
            if (result.length() > 0) {
                result = result.substring(0, result.length() - 2);
            }
            // Print the result
            System.out.println("Consecutive letters: " + result);
        }
        public static void main(String[] args) {
            Scanner sc = new Scanner(System.in);
            System.out.print("Enter a sentence: ");
            String sentence = sc.nextLine(); // Read the input sentence
            findConsecutiveLetters(sentence); // Call method to find
            consecutive letters
        }
    }
```

9.
```
    import java.util.*;
    class Sent_Merge_Rev
    {
        public static void main(String args[])
        {
            Scanner sc= new Scanner(System.in);
            System.out.print("Enter the number of sentences: "); //
            inputting the number of sentences to accept
            int n = sc.nextInt();
```

```
        String s = "";
        for(int i=1; i<=n; i++)
        {
            sc= new Scanner(System.in);
            System.out.print("Enter Sentence "+i+": ");
            s = s + sc.nextLine(); //inputting multiple sentences
            and joining them in the same String
        }
        //converting the sentence into StringTokenizer giving the
        given punctuation marks
        StringTokenizer str=new StringTokenizer(s," '.,;:!?");
        int c=str.countTokens();
        String w="", rev="";
        for(int i=1; i<=c; i++)
        {
            w = str.nextToken(); //extracting one word at a time
            rev = w+" "+rev; //joining the extracted words in reverse
            order
        }
        System.out.println("Output: "+rev);
    }
}
```

10.
```
import java.util.*;
public class SortWordsByLength {
    // Method to capitalize the first letter of a sentence
    public static String capitalizeFirstLetter(String sentence) {
        if (sentence == null || sentence.length() == 0) {
            return sentence;
        }
        return sentence.substring(0, 1).toUpperCase() + sentence.
        substring(1);
    }
    // Method to sort words by length and maintain original order
    for words of the same length
    public static String sortWordsByLength(String sentence)
```

```
{
    StringTokenizer st= new StringTokenizer(sentence," .");
    int count=st.countTokens();
    String starr[]=new String[count];
    int i=0,j;
    String temp,nstr="";
    while(st.hasMoreTokens())
    {
        String w=st.nextToken();
        starr[i++]=w;
    }
    for(i=0;i<starr.length-1;i++)
    {
        for(j=0;j<starr.length-1-i;j++)
        {
            if(starr[j].length()>starr[j+1].length())
            {
                temp=starr[j];
                starr[j]=starr[j+1];
                starr[j+1]=temp;
            }
        }
    }
    for(i=0;i<count;i++)
        nstr=nstr+starr[i]+" ";
    nstr=nstr+".";
    // Capitalize the first letter and add the full stop at the
    end
    return nstr;
}
public static void main(String[] args)
{
    Scanner sc = new Scanner(System.in);
    // Read the input sentence
```

```
        System.out.print("Enter a sentence: ");
        String sentence = sc.nextLine();
        // Sort the words by length and print the output
        String result = sortWordsByLength(sentence);
        System.out.println("Output: " + result);
    }
}
```

11.
```
import java.util.*;
class ReverseSentence {
    // Method to remove punctuation marks from the sentence
    public static String removePunctuation(String sentence) {
        String result = "";
        // Iterate over each character of the input string
        for (int i = 0; i < sentence.length(); i++) {
            char ch = sentence.charAt(i);
            // Check if the character is a letter, digit, or space
            (ignore punctuation)
           if ((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z')
                    || (ch >= '0' && ch <= '9') || ch == ' ') {
                result += ch; // Append to the result if it's not
                punctuation
            }
        }
        return result;
    }
    // Method to reverse the sentence
    public static String reverseSentence(String sentence) {
        // Use StringTokenizer to split sentence into words
        StringTokenizer st = new StringTokenizer(removePunctuation
        (sentence));
        int c = st.countTokens();
        String[] sarr = new String[c];
        String reversed = "";
        // Reverse the order of words
```

```
            while (st.hasMoreTokens()) {
                reversed = st.nextToken() + " " + reversed;  // Add words
                at the beginning
            }
            // Return the reversed sentence as a string after trimming
            the extra space
            return reversed.trim();
        }
        // Main method
        public static void main(String[] args) {
            Scanner sc = new Scanner(System.in);
            // Accept the input sentence
            System.out.print("Enter a sentence: ");
            String sentence = sc.nextLine();
            // Reverse the cleaned sentence
            String reversedSentence = reverseSentence(sentence);
            // Display the reversed sentence
            System.out.println("Reversed sentence without punctuation:
            " + reversedSentence);
        }
    }
```

12.
```
    import java.util.*;
    public class Frequency {
        // Data Members
        String mstr; // Original string
        String subs; // Substring to be counted
        // Member Method to input the strings
        void input() {
            Scanner sc = new Scanner(System.in);
            // Accepting the main string
            System.out.print("Enter the main string: ");
            mstr = sc.nextLine();
            // Accepting the substring to count
            System.out.print("Enter the substring to count: ");
            subs = sc.nextLine();
```

```
    }
    // Member Method to find the frequency of subs in mstr
    void find() {
        int count = 0; // Counter to store the frequency
        int index = 0; // Index for searching the substring
        // Convert both mstr and subs to lowercase for case-insensitive
        search
        mstr = mstr.toLowerCase();
        subs = subs.toLowerCase();
        // While substring is found in the main string
        while ((index = mstr.indexOf(subs, index)) != -1) {
            count++; // Increment count when substring is found
            index += subs.length(); // Move the index forward to
            avoid overlap
        }
        // Displaying the frequency
        System.out.println("Frequency of '" + subs + "' in the main
        string: " + count);
    }
    // Main function to run the program
    public static void main(String[] args) {
        Frequency obj = new Frequency();
        obj.input();  // Calling input method to accept the strings
        obj.find();   // Calling find method to count and display
        the frequency
    }
}
```

13.
```
import java.util.*;
class CaesarCipher
{
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);
        System.out.println("Enter plain text:");
        String str = in.nextLine();
        int len = str.length();
        if (len <= 3 || len >= 100) {
```

```
            System.out.println("INVALID LENGTH");
            return;
        }
        StringBuffer sb = new StringBuffer();
        for (int i = 0; i < len; i++) {
            char ch = str.charAt(i);
            if ((ch >= 'A' && ch <= 'M') || (ch >= 'a' && ch <= 'm'))
            {
                sb.append((char)(ch + 13));
            }
            else if ((ch >= 'N' && ch <= 'Z') || (ch >= 'n' && ch
            <= 'z')) {
                sb.append((char)(ch - 13));
            }
            else {
                sb.append(ch);
            }
        }
        String cipher = sb.toString();
        System.out.println("The cipher text is:");
        System.out.println(cipher);
    }
}
```

14.
```
import java.util.*;
class Program {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        // Accepting the input sentence
        System.out.println("Enter a sentence:");
        String str = sc.nextLine();
        str += " ";  // Add a space at the end to handle the last
        word easily
        int len = str.length();
        String word = "";
        int wordCount = 0;
        char ch;
        // Counting number of words
```

```
        for (int i = 0; i < len; i++) {
            if (str.charAt(i) == ' ') {
                wordCount++;
            }
        }
        // Creating an array to store words
        String wordArr[] = new String[wordCount];
        int index = 0;
        // Storing words in the array
        for (int i = 0; i < len; i++) {
            ch = str.charAt(i);
            if (ch != ' ') {
                word = word + ch;  // Build the current word
            } else {
                wordArr[index++] = word;  // Store the word and reset
                it
                word = "";
            }
        }
        // Bubble sort by word length
        for (int i = 0; i < wordCount - 1; i++) {
            for (int j = 0; j < wordCount - i - 1; j++) {
                if (wordArr[j].length() > wordArr[j + 1].length()) {
                    String t = wordArr[j];
                    wordArr[j] = wordArr[j + 1];
                    wordArr[j + 1] = t;
                }
            }
        }
        // Printing the words sorted by length
        System.out.println("Words sorted by length:");
        for (int i = 0; i < wordCount; i++) {
            System.out.print(wordArr[i] + " ");
        }
    }
}
```

15.
```
import java.util.*;
class Duplicate
 {
    // Data Members
    String sen;      // Original sentence
    String newsen;   // New sentence without duplicate characters
    // Constructor to initialise the data members
    Duplicate() {
        sen = "";
        newsen = "";
    }
    // Member Method to input the sentence
    void input() {
        Scanner sc = new Scanner(System.in);
        // Accept the sentence from the user
        System.out.print("Enter a sentence: ");
        sen = sc.nextLine();
    }
    // Member Method to remove duplicate characters
    void removedup() {
        // Create a boolean array to track if a character has been
        seen
        boolean[] seen = new boolean[256];  // ASCII characters
        // Traverse each character of the sentence
        for (int i = 0; i < sen.length(); i++) {
            char ch = sen.charAt(i);
            // If the character has not been seen before, add it to
            newsen
            if (!seen[ch]) {
                newsen += ch;
                seen[ch] = true;  // Mark the character as seen
            }
        }
    }
```

```
    // Member Method to display both the original and new sentence
    void show() {
        System.out.println("Original Sentence: " + sen);
        System.out.println("New Sentence without duplicates: " +
        newsen);
    }
    public static void main(String[] args) {
        // Create an object of the class
        Duplicate obj = new Duplicate();
        // Call the member methods
        obj.input();       // Input the sentence
        obj.removedup();   // Remove duplicate characters
        obj.show();        // Display both the sentences
    }
}
```

11. Basic Input/Output

Unsolved Questions

A. 1. a 2. d 3. a 4. c 5. c

B. 1. a. i>n b. n%i==0 c. i d. factors(n,i+1) e. factors(n,i+1)

2. a. lc % a == 0 && lc % b == 0
   b. System.out.println(lc)
   c. lcm(a, b, lc + 1)

3. a. p<0 b. return 0 c. "AEIOUaeiou".indexOf(s.charAt(p)) != -1
   d. count(s, p - 1); e. return count(s, p - 1);

4. a. n!=0 b. n%2==0 c. n*3+1

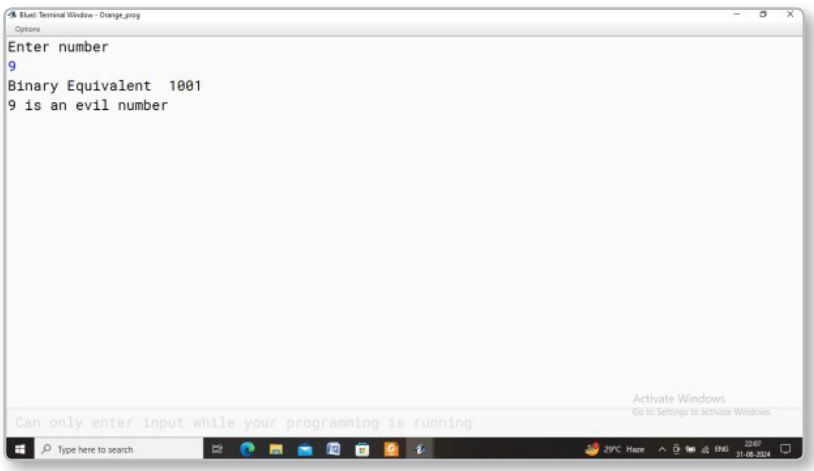
C. 1. i. JUNE
   JNUE
   ii. SCROLL
   LCROLS
   LLROCS
   LLORCS

2. return 36

3. Output
   1
   2
   3
   4
   5
4. i. 1
   ii. 0
   iii. It is checking if 'n' is a prime number or not
5. The method is returning 55
6. The method will return
   {{}, {{}}, {{}, {{}}}}
7.
```
import java.util.*;
class Evilnum
{ int num;
  void getnum()
  { Scanner sc=new Scanner(System.in);
    System.out.println("Enter number");
    num=sc.nextInt();
  }
  String tobinary(int a)
  { if(a==0)
     return " ";
    else
    { return tobinary(a/2)+Integer.toString(a%2);}
  }
  int countones(String s)
  { if(s.equals(""))
      return 0;
    else
    { if(s.charAt(0)=='1')
        return 1+countones(s.substring(1));
      else
        return countones(s.substring(1));
    }}
   void isEvil()
   { String s=tobinary(num);
```

```
        System.out.println("Binary Equivalent "+s);
        int c=countones(s);
        if(c%2==0)
          System.out.println(num+" is an evil number");
        else
          System.out.println(num+" is not an evil number");
      }
      public static void main(String[] args)
      { Evilnum ob=new Evilnum();
        ob.getnum();
        ob.isEvil();
      }}
```

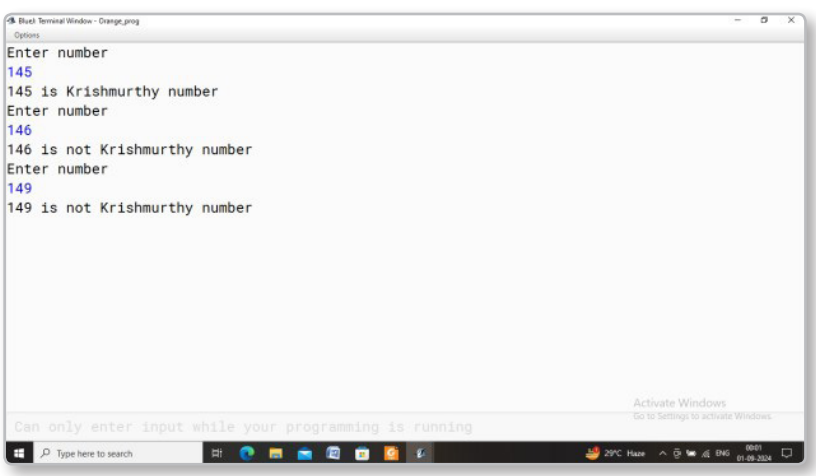
Output:



8.
```
import java.util.*;
 class Kmurty
 {
  int num;
  void getnum()
  { Scanner sc=new Scanner(System.in);
    System.out.println("Enter number");
    num=sc.nextInt();
  }
  int factorial(int a)
  { if(a==1)
      return 1;
```

```
    else
      return a*factorial(a-1);
    }
  int sumfact(int x)
  { if(x==0)
      return 0;
    else
      return factorial(x%10)+sumfact(x/10);
    }
    void check()
    { if(sumfact(num)==num)
        System.out.println(num+" is Krishmurthy number");
      else
        System.out.println(num+" is not Krishmurthy number");
    }
    public static void main(String[] args)
    { Kmurty ob=new Kmurty();
      ob.getnum();
      ob.check();
    }}
```

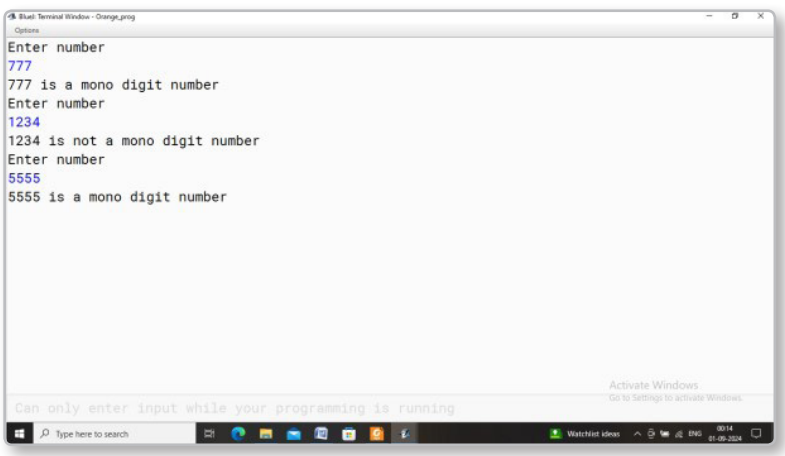
Output:



9.
```
import java.util.*;
class Mono
{ int num;
  void getnum()
```

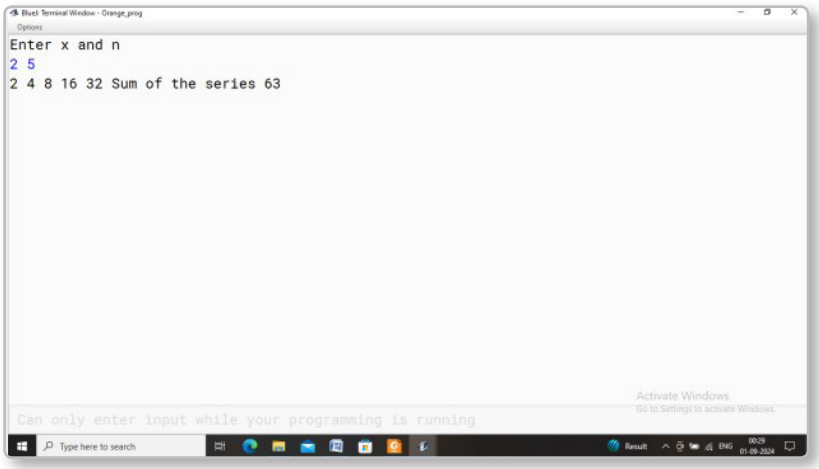
```
  { Scanner sc=new Scanner(System.in);
    System.out.println("Enter number");
    num=sc.nextInt();
  }
  boolean check(int n,int d)
  { if(n==0)
      return true;
    else
    { if(n%10!=d)
        return false;
      else
        return check(n/10,d);
    }
 }
 void print()
 { if(check(num,num%10))
     System.out.println(num+" is a mono digit number");
   else
     System.out.println(num+" is not a mono digit number");
    }
  public static void main(String[] args)
  { Mono ob=new Mono();
    ob.getnum();
    ob.print();
}}
```

Output:



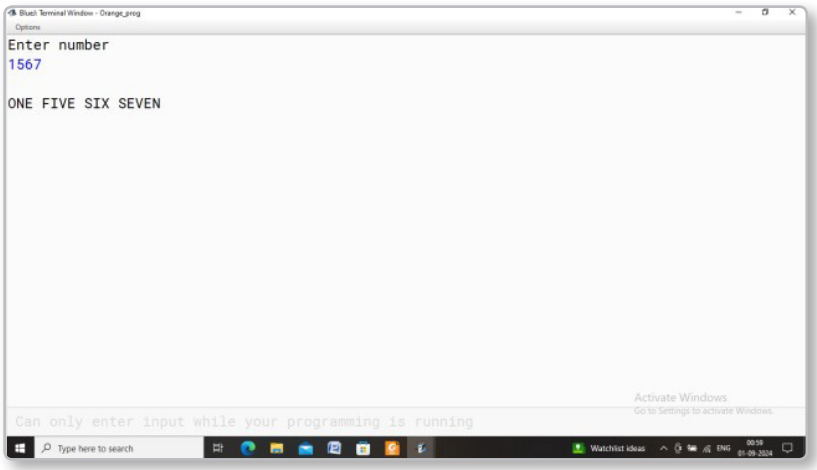
10.
```
import java.util.*;
 class CalSeries
 { int x,n;
   CalSeries()
   { x=n=0;}
   void input()
   { Scanner sc=new Scanner(System.in);
     System.out.println("Enter x and n");
     x=sc.nextInt();
     n=sc.nextInt();
    }
   int power(int p,int q)
   { if(q==0)
       return 1;
     else
       return p*power(p,q-1);
    }
    void cal()
    { int s=1,t;
      for(int i=1;i<=n;i++)
      { t=power(x,i);
        System.out.print(t+" ");
        s=s+t;
      }
      System.out.println("Sum of the series "+s);
    }
    public static void main(String[] args)
    { CalSeries ob=new CalSeries();
      ob.input();
      ob.cal();
    }}
```

Output:



11.
```
import java.util.*;
class ToWord
{ int num;
  ToWord()
  { num=0;}
   void input()
   { Scanner sc=new Scanner(System.in);
     System.out.println("Enter number");
     num=sc.nextInt();
     print(num);
    }
   void print(int n)
   { String a[]={"ZERO","ONE","TWO","THREE","FOUR","FIVE","SIX","SEVEN","EIGHT","NINE"};
     if(n==0)
       System.out.println();
     else
       { print(n/10);
         System.out.print(a[n%10]+" ");
        }
    }
    public static void main(String[] args)
    { ToWord ob=new ToWord();
      ob.input();
    }}
```

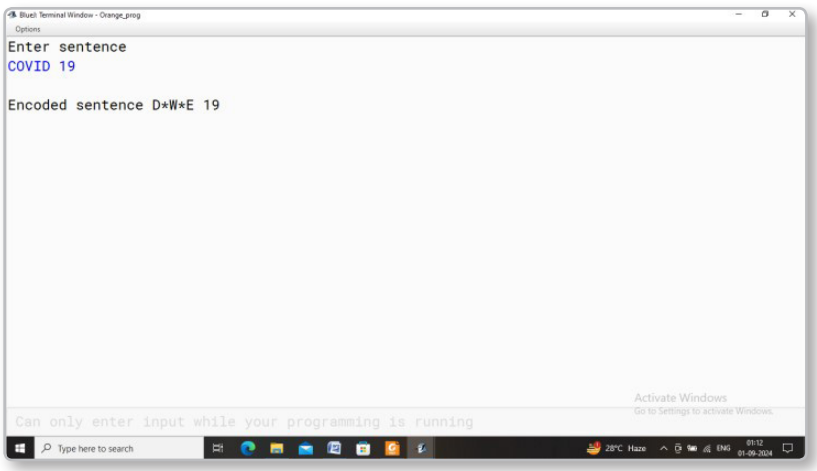
Output:



12.
```
import java.util.*;
class Encode
{ String sen,esen;
  Encode()
  { sen=esen="";}
  void input()
  { Scanner sc=new Scanner(System.in);
    System.out.println("Enter sentence");
    sen=sc.nextLine();
    change(sen,0);
    System.out.println("Encoded sentence "+esen);
  }
  void change(String s,int p)
  { if(p==s.length())
      System.out.println();
    else
    { if("AEIOU".indexOf(s.charAt(p))!=-1)
        esen=esen+"*";
      else
      { if(Character.isLetter(s.charAt(p)))
        esen=esen+(char)((int)s.charAt(p)+1);
        else
        esen=esen+s.charAt(p);
```

```
            }
            change(s,p+1);
}}
    public static void main(String[] args)
    { Encode ob=new Encode();
      ob.input();
    }}
```

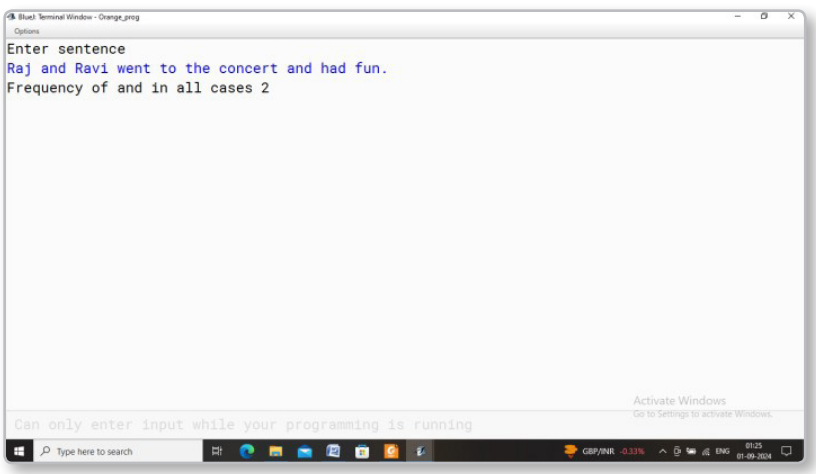
Output:



13.
```
import java.util.*;
class Count
{ String line;
  Count()
  { line="";}
  void input()
  { Scanner sc=new Scanner(System.in);
    System.out.println("Enter sentence");
    line=sc.nextLine();
    line=line+" ";
  }
  int cal(String s)
  { if(s.equals(""))
      return 0;
    else
    { String w=s.substring(0,s.indexOf(" "));
```

```
        if(w.equalsIgnoreCase("And"))
          return 1+cal(s.substring(s.indexOf(" ")+1));
        else
          return cal(s.substring(s.indexOf(" ")+1));
      }
}
void print()
{ int c=cal(line);
  System.out.println("Frequency of and in all cases "+c);
}
public static void main(String[] args)
{ Count ob=new Count();
  ob.input();
  ob.print();
}}
```

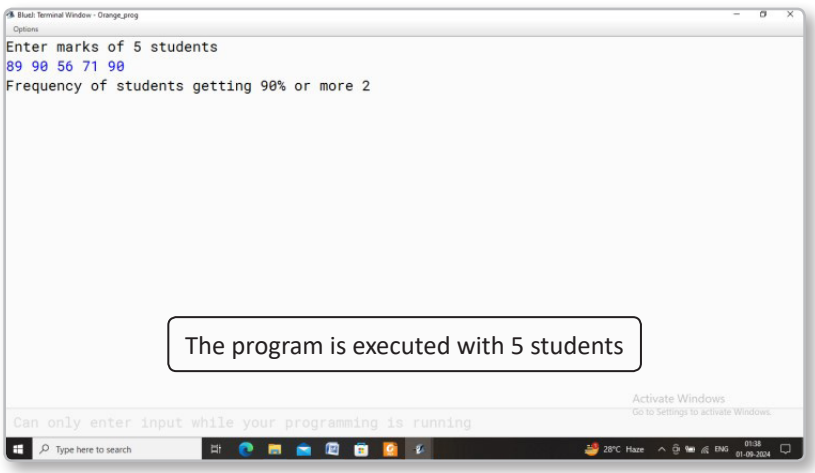
Output:



14.
```
import java.util.*;
class Marks
{ int mark[]=new int[50];
  Marks()
  { for(int i=0;i<50;i++)
    { mark[i]=0;}
  }
  void input()
```

```
  { Scanner sc=new Scanner(System.in);
    System.out.println("Enter marks of 50 students");
    for(int i=0;i<50;i++)
    { mark[i]=sc.nextInt();}
  }
  int count(int p)
  { if(p==mark.length)
      return 0;
    else
    { if(mark[p]>=90)
        return 1+count(p+1);
      else
        return count(p+1);
    }
}
void print()
{ int c=count(0);
  System.out.println("Frequency of students getting 90% or more "+c);
}
public static void main(String[] args)
{ Marks ob=new Marks();
  ob.input();
  ob.print();
}}
```

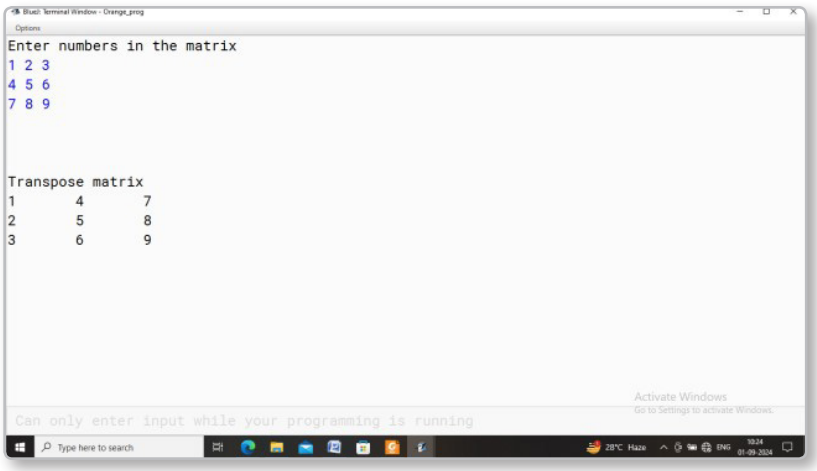
Output:



15.
```
import java.util.*;
class Transpose {
    int s = 3, a[][], b[][];  // Fixed size of 3x3 matrix
    // Constructor to initialise the matrix size and matrices a and b
    Transpose() {
        a = new int[s][s];  // Matrix to store input values
        b = new int[s][s];  // Matrix to store the transpose
    }
    // Method to input matrix values
    void input() {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter numbers in the matrix:");
        for (int i = 0; i < s; i++) {
            for (int j = 0; j < s; j++) {
                a[i][j] = sc.nextInt();
            }
        }
    }
    // Method to generate the transpose recursively
    void generate(int r, int c) {
        if (r == s) {
            return;  // Base case: When row index reaches the matrix
            size
        } else {
            if (c == s) {
                c = 0;
                r = r + 1;  // Move to the next row
            }
            if (r < s) {
                b[r][c] = a[c][r];   // Transpose: Swap rows and
                columns
                generate(r, c + 1);  // Recur for the next column
            }
        }
```

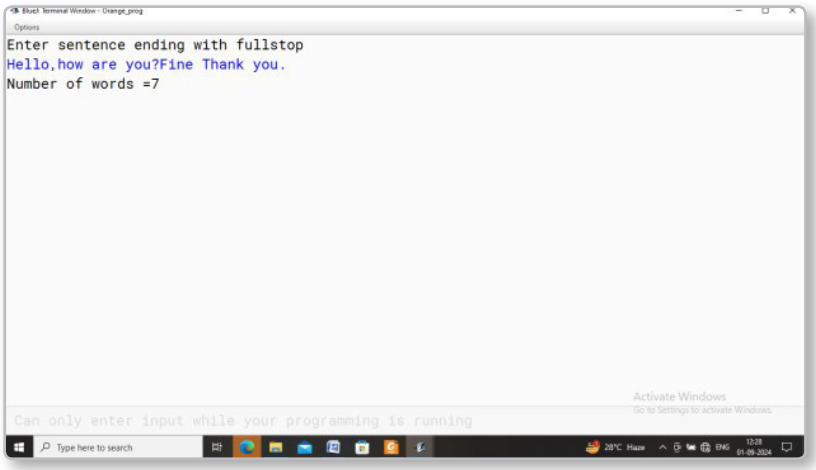
```
    }
    // Method to print the transpose matrix
    void print() {
        generate(0, 0);  // Start recursion from row 0, column 0
        System.out.println("Transpose matrix:");
        for (int i = 0; i < s; i++) {
            for (int j = 0; j < s; j++) {
                System.out.print(b[i][j] + "\t");  // Print transpose
                matrix
            }
            System.out.println();
        }
    }
    // Main method
    public static void main(String[] args) {
        // Create object of Transpose class
        Transpose ob = new Transpose();
        ob.input();  // Accept the matrix
        ob.print();   // Print the transpose of the matrix
    }
}
```

Output:



16.
```
import java.util.*;
class Word
{ String s;
  int len;
  void read()
  { Scanner sc=new Scanner(System.in);
    System.out.println("Enter sentence ending with fullstop");
    s=sc.nextLine();
    len=s.length();
  }
  int count(int l)
  { if(l==len)
      return 0;
    else if(s.charAt(l)==' '||s.charAt(l)==','||s.charAt(l)=='?'
    || s.charAt(l)=='.')
       return 1+count(l+1);
    else
       return count(l+1);
   }
   void show()
   { int c=count(0);
     System.out.println("Number of words ="+c);
    }
    public static void main(String[] args)
    { Word ob=new Word();
      ob.read();
      ob.show();
    }
}
```

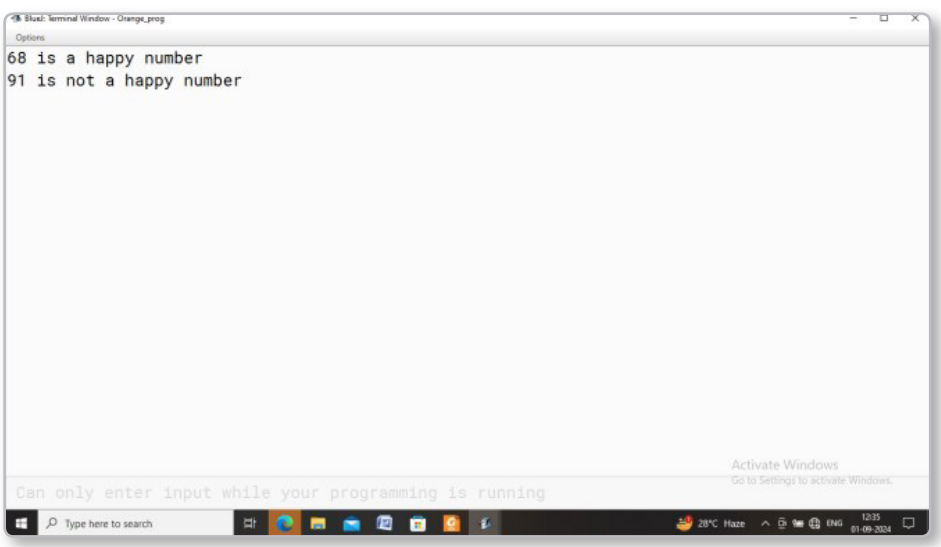
Output:



17.
```
import java.util.*;
class Happy {
    int n;
    // Constructor to initialise n to 0
    Happy() {
        n = 0;
    }
    // Method to set the number n
    void getnum(int nn) {
        n = nn;
    }
    // Method to compute the sum of squares of digits
    int sumSqdigits(int x) {
        if (x == 0)
            return 0;
        else {
            return (x % 10) * (x % 10) + sumSqdigits(x / 10);
        }
    }
    // Method to check if the number is a happy number
    void isHappy() {
        int a = n;
```

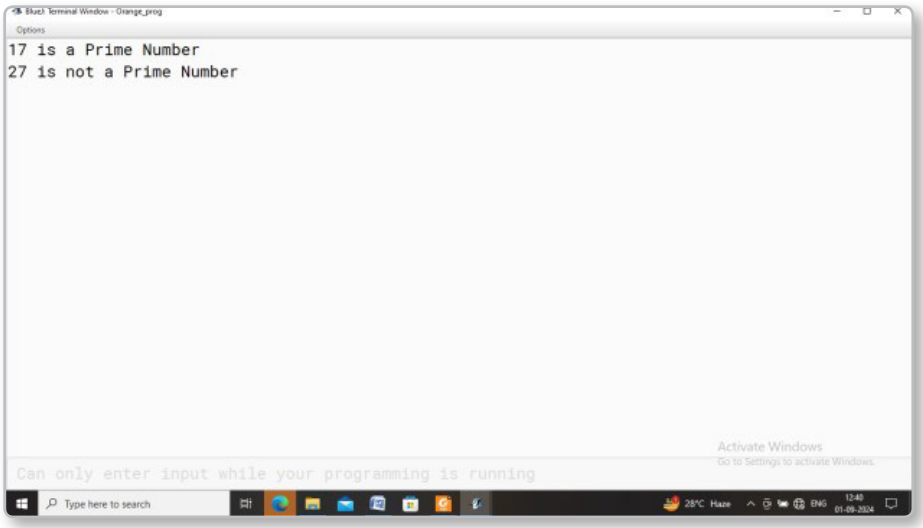
```
        while (a > 9) {
            a = sumSqdigits(a);
        }
        if (a == 1)
            System.out.println(n + " is a happy number");
        else
            System.out.println(n + " is not a happy number");
    }
    // Main method
    public static void main(String[] args) {
        // Get the input from the user
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int a = sc.nextInt();  // Read the number from user input
        // Create an object of the Happy class
        Happy ob = new Happy();
        ob.getnum(a);  // Set the number in the object
        ob.isHappy();  // Check if it's a happy number
    }
}
```

Output:



18.
```
import java.util.*;
class Prime {
    int num; // member data
    // Parameterised constructor to initialise 'num'
    Prime(int nn) {
        num = nn;
    }
    // Method to count factors of the number 'num'
    int countfactors(int i) {
        if (i > num) // base case: if i exceeds num, return 0
            return 0;
        else if (num % i == 0) // if i is a factor of num
            return 1 + countfactors(i + 1); // increment count
        else
            return countfactors(i + 1); // recursively check next
            value of i
    }
    // Method to check if the number is prime
    void check() {
        if (countfactors(1) == 2) // if the count of factors is 2
        (prime)
            System.out.println(num + " is a Prime Number");
        else
            System.out.println(num + " is not a Prime Number");
    }
    // Main method to run the program
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int x = sc.nextInt();  // Read the input number
        Prime obj = new Prime(x); // Create an object of Prime class
        with x
        obj.check(); // Check if the number is prime
    }
}
```

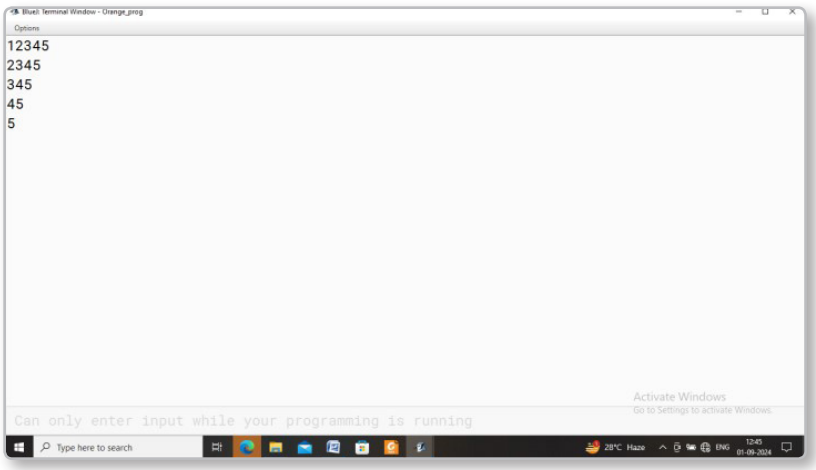
Output:



19.
```
import java.util.*;
class Pattern {
    int n;
    // Method to read the value of n
    void read(int nn) {
        n = nn;
    }
    // Method to print the pattern recursively
    void print(int i, int j) {
        if (i > n)
            System.out.println();
        else if (j > n) {
            System.out.println();
            print(i + 1, i + 1); // Move to the next row and reset
            column to i+1
        } else {
            System.out.print(j + " "); // Print the value of j
            print(i, j + 1); // Move to the next column
        }
    }
```

```
    // Main method to run the program
    public static void main(String[] args) {
        // Create an object of Pattern class
        Pattern ob = new Pattern();
        // Create Scanner object to read input from the user
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int x = sc.nextInt();  // Read the input number
        // Initialise the pattern object with x
        ob.read(x);
        // Print the pattern
        ob.print(1, 1);
    }
}
```

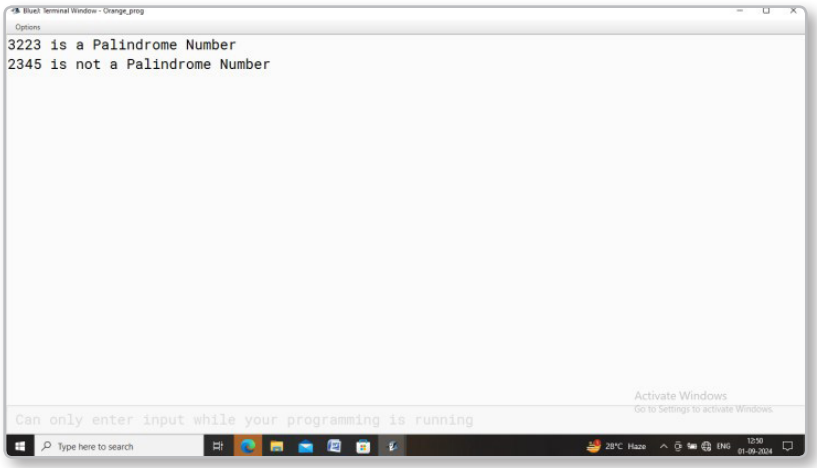
Output:



20.
```
import java.util.*;
class Palindrome {
    int num, rev; // member data
    // Parameterised constructor to initialise num and rev
    Palindrome(int nn) {
        num = nn;
        rev = 0;
    }
```

```
    // Recursive method to calculate the reverse of the number
    int revnum(int i) {
        if (i == 0) // base case: when i becomes 0, return the
        reversed number
            return rev;
        else {
            rev = rev * 10 + i % 10; // Add the last digit to rev
            return revnum(i / 10); // Recursive case: process the
            rest of the number
        }
    }
    // Method to check if the number is a palindrome
    void check() {
        if (revnum(num) == num) // If number equals its reverse,
        it's a palindrome
            System.out.println(num + " is a Palindrome Number");
        else
           System.out.println(num + " is not a Palindrome Number");
    }
    // Main method
    public static void main(String[] args) {
        // Create a scanner to take user input
        Scanner sc = new Scanner(System.in);
        // Read the number to check
        System.out.print("Enter a number: ");
        int x = sc.nextInt();  // Take input from the user
        // Create an object of Palindrome with the input number
        Palindrome obj = new Palindrome(x);
        // Call the check method to see if it's a palindrome
        obj.check();
    }
}
```

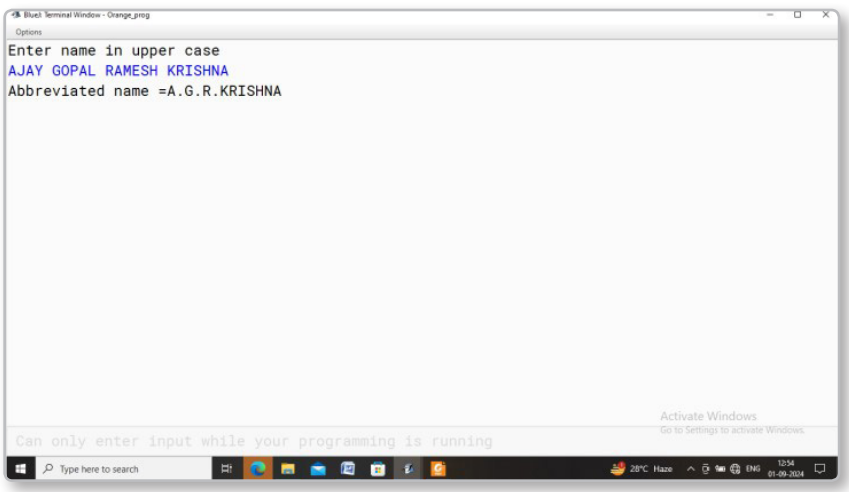
Output:



21.
```
import java.util.*;
class Abword
{ String nm,anm;
  void read() //accept
  { Scanner sc=new Scanner(System.in);
    System.out.println("Enter name in upper case");
    nm=sc.nextLine();
    nm=" "+nm;
    anm="";
   }
  String print(String a)
  {
    if(a.equals("")) // end of name
      return anm;
   else if(a.charAt(0)==' ') // if space
   { anm=anm+a.charAt(1)+"."; // add next character
     return print(a.substring(1)); // recursive case
    }
    else
    { return print(a.substring(1));// recursive case
    }
  }
    void show() // display
```

```
    { int p=nm.lastIndexOf(' ');// last space position
      String s=nm.substring(0,p);
      String r=print(s); //passing name excluding surname
      r=r+nm.substring(p+1);
      System.out.println("Abbreviated name ="+r);
     }
     public static void main(String[] args)
     { Abword ob=new Abword();
       ob.read();
       ob.show();
     }
}
```

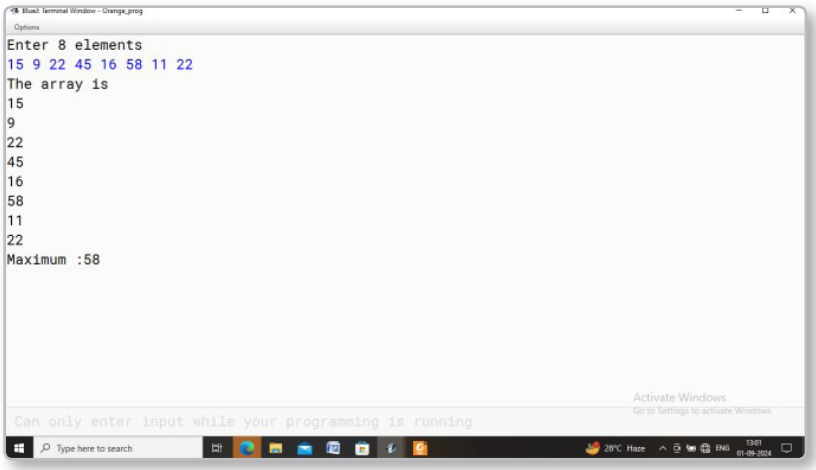
Output:



22.
```
import java.util.*;
class Maxnum {
     int arr[];
     int n, max;
     Maxnum(int nn) {
          n = nn;
          arr = new int[n];
     }
     void fillarray() {
```

```
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter " + n + " elements");
        for (int i = 0; i < n; i++)
            arr[i] = sc.nextInt();
        max = arr[0];
    }
    int maximum(int i) {
        if (i == n)
            return max;
        else {
            max = Math.max(max, arr[i]);
            return maximum(i + 1);
        }
    }
    void print() {
        System.out.println("The array is:");
        for (int i = 0; i < n; i++) {
            System.out.println(arr[i]);
        }
        int m = maximum(0);
        System.out.println("Maximum: " + m);
    }
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter size of array: ");
        int s = sc.nextInt();
        Maxnum obj = new Maxnum(s);
        obj.fillarray();
        obj.print();
    }
}
```

Output:

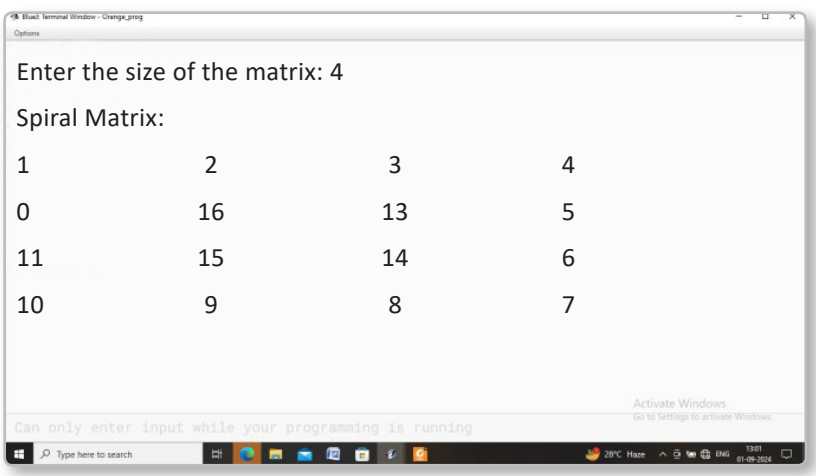


23.
```
import java.util.*;
class Spiral {
    int a[][]; // Two-dimensional matrix
    int n;     // Size of the matrix
    // Method to accept size of matrix and initialise the array
    void accept() {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the size of the matrix: ");
        n = sc.nextInt(); // Accept the matrix size
        a = new int[n][n]; // Initialise the matrix with size n x n
    }
    // Recursive method to generate the spiral matrix
    void generate(int rowpos, int colpos, int value, int size, int
    upper, int lower) {
        if (value <= n * n) {
            if (size == 1) { // Filling the lower row
                if (colpos != upper + 1) {
                    a[lower][colpos] = value;
                    generate(lower, colpos + 1, value + 1, 1, upper,
                    lower);
                } else {
                    generate(lower + 1, upper, value, 2, upper,
                    lower);
```

```
            }
        } else if (size == 2) { // Filling the right column
            if (rowpos != upper + 1) {
                a[rowpos][upper] = value;
                generate(rowpos + 1, colpos, value + 1, 2, upper,
                lower);
            } else {
                generate(upper, upper - 1, value, 3, upper,
                lower);
            }
        } else if (size == 3) { // Filling the upper row
            if (colpos != lower - 1) {
                a[upper][colpos] = value;
                generate(rowpos, colpos - 1, value + 1, 3, upper,
                lower);
            } else {
                generate(upper - 1, lower, value, 4, upper,
                lower);
            }
        } else if (size == 4) { // Filling the left column
            if (rowpos != lower + 1) {
                a[rowpos][lower] = value;
                generate(rowpos - 1, colpos, value + 1, 4, upper,
                lower);
            } else {
                generate(upper - 1, lower + 1, value, 1, upper
                - 1, lower + 1);
            }
        }
    }
}
// Method to display the matrix after generating the spiral
void display() {
    generate(0, 0, 1, 1, n - 1, 0); // Start recursive generation
    from (0, 0)
```

```
            System.out.println("Spiral Matrix:");
            for (int i = 0; i < n; i++) {
                for (int j = 0; j < n; j++) {
                    System.out.print(a[i][j] + "\t");   // Print each
                    element of the matrix
                }
                System.out.println();
            }
        }
        // Main method to run the program
        public static void main(String[] args) {
            Spiral ob = new Spiral();  // Create an object of the Spiral
            class
            ob.accept();  // Accept the matrix size
            ob.display();  // Display the spiral matrix
        }
}
```

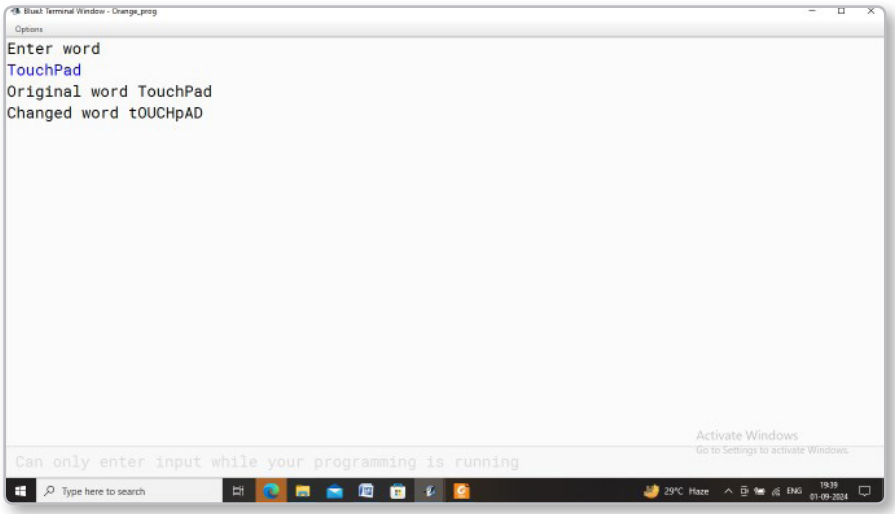
Output:



24.
```
import java.util.*;
class Change
{ String str,newstr;
  int len;
  Change()
```

```
  { str=newstr="";
    len=0;
  }
  void inputword()
  { Scanner sc=new Scanner(System.in);
    System.out.println("Enter word");
    str=sc.next();
    len=str.length();
  }
  char caseconvert(char c)
  { if(Character.isUpperCase(c))
      return Character.toLowerCase(c);
    else
      return Character.toUpperCase(c);
  }
  void recchange(int l)
  { if(l==len)
    {}
    else
    { newstr=newstr+=caseconvert(str.charAt(l));
      recchange(l+1);
    }}
   void display()
   { recchange(0);
     System.out.println("Original word "+str);
     System.out.println("Changed word "+newstr);
    }
    public static void main(String[] args)
    { Change ob=new Change();
      ob.inputword();
      ob.display();
    }
}
```

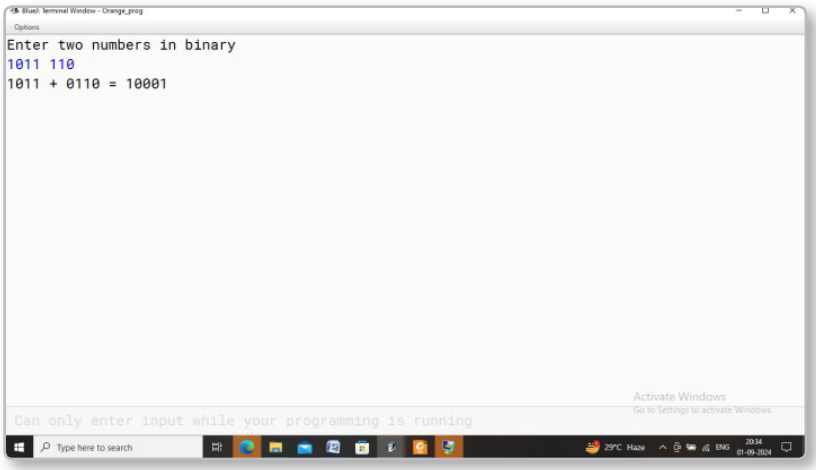
Output:



25.
```
import java.util.*;
class BinAdd
{ String num1,num2,res;
  int c;
  BinAdd()
  { num1=num2=res="";
    c=0;}
  void accept()
  { Scanner sc=new Scanner(System.in);
    System.out.println("Enter two numbers in binary ");
    num1=sc.next();
    num2=sc.next();
  }
  void addzero()
  { if(num1.length()>num2.length())
    { for(int i=1;i<=num1.length()-num2.length();i++)
        num2="0"+num2;
    }
    else
    {
     for(int i=1;i<=num2.length()-num1.length();i++)
```

```
        num1="0"+num1;
    } }
  void addnum(int p)
  { int c1=0,c2=0,s=0;
    if(p<0)
    {res=Integer.toString(c)+res;}
    else
    { c1=(num1.charAt(p)-48);
      c2=(num2.charAt(p)-48);
      s=c1+c2+c;
      if(s<=1)
      { c=0;}
      else if(s==2)
      { s=0;c=1;}
      else if(s==3)
      { s=1;c=1;}
      else{}
      res=Integer.toString(s)+res;
      addnum(p-1);
    }}
    void display()
    { addnum(num1.length()-1);
      System.out.println(num1+" + "+num2+" = "+res);
    }
    public static void main(String[] args)
    { BinAdd ob=new BinAdd();
      ob.accept();
      ob.addzero();
      ob.display();
    }
}
```

Output:



12. Inheritance, Interface and Polymorphism

Unsolved Questions

A. 1. a 2. b 3. d 4. c 5. a 6. d
7. a 8. c 9. b 10. a

B. 1. extends 2. Java 3. super() 4. protected 5. public
6. Hybrid Inheritance 7. Method Overloading 8. inherited
9. interface 10. Abstract

C. 1.
```
class Sale
{ protected int empno;
  protected String empName;
  protected double saleamt;
  Sale(int em,String en,double s)
  { empno=em;
    empName=en;
    saleamt=s;
  }
void display()
{ System.out.println("Employee number "+empno);
  System.out.println("Employee name "+empName);
```

```
  System.out.println("Sale amount "+saleamt);
}}

class Commission extends Sale
{  double com;
   Commission(int em,String en,double s)
   { super(em,en,s);
     com=0;
    }
 void calculate()
 { if(saleamt<=50000)
     com=saleamt*0.10;
   else if(saleamt<=1000000)
     com=saleamt*0.175;
   else if(saleamt<=1500000)
     com=saleamt*0.20;
   else
     com=saleamt*0.25;
 }
 void display()
 { super.display();
   System.out.println("Commission "+com);
}
```

2.
```
class Student
{ protected int sroll;
  protected String sname;
  protected double mark[]=new double[5];
  Student(int r,String n,double m[])
  { sroll=r;
    sname=n;
    for(int i=0;i<5;i++)
    { mark[i]=m[i];}
  }
void display()
```

```
{ System.out.println("Roll "+sroll+" Name "+sname);
  System.out.println("Marks in 5 subjects ");
  for(int i=0;i<5;i++)
  { System.out.print(mark[i]+"\t");}
}
}
class Grade extends Student
{ double avg;
  char grade;
  Grade(int r,String s,double m[])
  { super(r,s,m);
    avg=0.0;
    grade="";
  }
void calculate()
{ double s=0;
  for(int i=0;i<5;i++)
  { s=s+mark[i];}
  avg=s/5;
  if(avg>=90)
    grade="A";
  else if(avg>=75)
    grade="B";
  else if(avg>=60)
    grade="C";
  else if(avg>=40)
    grade="D";
  else
    grade="E";
}
void display()
{ super.display();
  System.out.println("\n Average "+avg+" Grade "+grade);
}
```

3.
```
class Array2D
{
    protected int a[][] = new int[3][3];
    Array2D(int c[][])
    {
        for (int i = 0; i < 3; i++)
        {
            for (int j = 0; j < 3; j++)
            {
                a[i][j] = c[i][j];
            }
        }
    }
    void print()
    {
        System.out.println("Original matrix");
        for (int i = 0; i < 3; i++)
        {
            for (int j = 0; j < 3; j++)
            {
                System.out.print(a[i][j] + "\t");
            }
            System.out.println();
        }
    }
}
class Transpose extends Array2D
{
    int b[][] = new int[3][3];
    Transpose(int c[][])
    {
        super(c);
        for (int i = 0; i < 3; i++)
        {
```

```
            for (int j = 0; j < 3; j++)
            {
                b[i][j] = 0; // Though not needed, kept as you had
                written.
            }
        }
    }
    void generate()
    {
        for (int i = 0; i < 3; i++)
        {
            for (int j = 0; j < 3; j++)
            {
                b[j][i] = a[i][j];
            }
        }
    }
    void print()
    {
        super.print();
        System.out.println("\nTranspose Matrix");
        for (int i = 0; i < 3; i++)
        {
            for (int j = 0; j < 3; j++)
            {
                System.out.print(b[i][j] + "\t");
            }
            System.out.println();
        }
    }
}
```

4.
```
class Array2D
{
    protected int a[][] = new int[3][3];
```

```
    Array2D(int c[][])
    {
        for (int i = 0; i < 3; i++)
        {
            for (int j = 0; j < 3; j++)
            {
                a[i][j] = c[i][j];
            }
        }
    }
    void print()
    {
        System.out.println("Original matrix");
        for (int i = 0; i < 3; i++)
        {
            for (int j = 0; j < 3; j++)
            {
                System.out.print(a[i][j] + "\t");
            }
            System.out.println();
        }
    }
}
class Upper extends Array2D
{
    int b[][] = new int[3][3];
    Upper(int c[][])
    {
        super(c);
        for (int i = 0; i < 3; i++)
        {
            for (int j = 0; j < 3; j++)
            {
                b[i][j] = a[i][j];
```

```
            }
        }
    }
    void generate()
    {
        for (int i = 0; i < 3; i++)
        {
            for (int j = 0; j < 3; j++)
            {
                if (i > j)
                    b[i][j] = 0;
            }
        }
    }
    void print()
    {
        super.print();
        System.out.println("\nUpper Triangular Matrix");
        for (int i = 0; i < 3; i++)
        {
            for (int j = 0; j < 3; j++)
            {
                System.out.print(b[i][j] + "\t");
            }
            System.out.println();
        }
    }
}
```

5.
```
class Word
{
    protected String str;
    protected int len;
    Word(String s)
    {
```

```
        str = s;
        len = str.length();
    }
    void display()
    {
        System.out.println("Original Word " + str);
    }
}
class ChangeCase extends Word
{
    String nstr;
    ChangeCase(String s)
    {
        super(s);
        nstr = "";
    }
    void change()
    {
        for (int i = 0; i < len; i++)
        {
            char c = str.charAt(i);
            if (Character.isUpperCase(c))
            {
                nstr = nstr + Character.toLowerCase(c);
            }
            else if (Character.isLowerCase(c))
            {
                nstr = nstr + Character.toUpperCase(c);
            }
            else
            {
                nstr = nstr + c;
            }
        }
```

```
        }
        void display()
        {
            super.display();
            System.out.println("Changed Word " + nstr);
        }
    }
```

6.
```
    class Quadratic
    {
        double a, b, c, d;
        Quadratic(double a1, double b1, double c1)
        {
            a = a1;
            b = b1;
            c = c1;
        }
        void calculate()
        {
            d = b * b - 4 * a * c;
        }
        void display()
        {
            System.out.println("a=" + a + " b=" + b + " c=" + c + " d="
            + d);
        }
    }
    class Roots extends Quadratic
    {
        double r1, r2;
        Roots(double a1, double b1, double c1)
        {
            super(a1, b1, c1);
            r1 = r2 = 0.0;
        }
        void calculate()
```

```
    {
        super.calculate();
        if (d < 0)
            System.out.println("Roots are imaginary");
        else if (d == 0)
            System.out.println("Roots are real and equal");
        else
            System.out.println("Roots are real and unequal");
    }
    void display()
    {
        super.display();
      r1=-b+Math.sqrt(d)/(2*a);
      r2=-b-Math.sqrt(d)/(2*a);
        System.out.println("Roots are " + r1 + " and " + r2);
    }
}
```

7.
```
class Detail {
    protected String name, address, telno;
    protected double call, rent;
    Detail(String n, String a, String t, double c, double r) {
        name = n;
        address = a;
        telno = t;
        call = c;
        rent = r;
    }
    void show() {
        System.out.println("Name " + name + " Address " + address +
        " Telephone no " + telno);
        System.out.println("No. of calls " + call + " Rental charge
        " + rent);
    }
}
class Bill extends Detail {
```

```
    double amt;
    Bill(String n, String a, String t, double c, double r) {
        super(n, a, t, c, r);
        amt = 0.0;
    }
    void cal() {
        if (call <= 100)
            amt = rent;
        else if (call <= 200)
            amt = call * 0.6 + rent;
        else if (call <= 300)
            amt = call * 0.8 + rent;
        else
            amt = call * 1 + rent;
    }
    void show() {
        super.show();
        System.out.println("Bill amount " + amt);
    }
}
```

8.
```
class Worker {
    protected String name;
    protected double basic;
    Worker(String n, double b) {
        name = n;
        basic = b;
    }
    void display() {
        System.out.println("Name " + name);
        System.out.println("Basic " + basic);
    }
}
class Wages extends Worker {
    int hrs;
    double rate, wage;
```

```
    Wages(String n, double b, int h, double r) {
        super(n, b);
        hrs = h;
        rate = r;
    }
    double overtime() {
        return hrs * rate;
    }
    void display() {
        wage = basic + overtime();
        super.display();
        System.out.println("Hours worked " + hrs + " Rate " + rate
        + " Total wage " + wage);
    }
}
```

9.
```
class Electric {
    protected String name;
    protected int consno, pres_read, pre_read, unit;
    Electric(String n, int c, int pr, int pv) {
        name = n;
        consno = c;
        pres_read = pr;
        pre_read = pv;
        unit = pres_read - pre_read;
    }
    void show() {
        System.out.println("Name " + name + " Consumer number " +
        consno);
        System.out.println("Previous reading " + pre_read + " Present
        reading " + pres_read + " Units consumed " + unit);
    }
}
class Bill extends Electric {
    double amt;
    Bill(String n, int c, int pr, int pv) {
```

```
        super(n, c, pr, pv);
        amt = 0.0;
    }
    void calculate() {
        if (unit <= 100)
            amt = 250.0;
        else if (unit <= 300)
            amt = 250.0 + (unit - 100) * 1;
        else
            amt = 250.0 + 200 * 1 + (unit - 300) * 1.5;
    }
    void show() {
        super.show();
        System.out.println("Bill amount " + amt);
    }
}
```

10.
```
class Money {
    protected int rs1, ps1;
    Money(int r1, int p1) {
        rs1 = r1;
        ps1 = p1;
    }
    void show() {
        System.out.println("First amount Rs " + rs1 + " " + ps1 + "
        paisa");
    }
}
class AddMoney extends Money {
    int rs2, ps2, totrs, totps;
    AddMoney(int r1, int p1, int r2, int p2) {
        super(r1, p1);
        rs2 = r2;
        ps2 = p2;
        totrs = totps = 0;
```

```
    }
    void addamt() {
        totrs = rs1 + rs2;
        totps = ps1 + ps2;
        if (totps > 100) {
            totps -= 100;
            totrs++;
        }
    }
    void show() {
        super.show();
        System.out.println("Second amount Rs " + rs2 + " " + ps2 +
        " paisa");
        System.out.println("Final amount Rs " + totrs + " " + totps
        + " paisa");
    }
}
```

11.
```
class Student {
    protected int sroll;
    protected String sname;
    protected int mark[] = new int[5];
    Student(int sr, String sn, int m[]) {
        sroll = sr;
        sname = sn;
        for (int i = 0; i < 5; i++) {
            mark[i] = m[i];
        }
    }
    void display() {
        System.out.println("Roll " + sroll + " Name " + sname);
        System.out.println("Marks in five subjects");
        for (int i = 0; i < 5; i++) {
            System.out.print(mark[i] + "\t");
        }
        System.out.println();
```

```
    }
}
class BestFour extends Student {
    double tot, avg;
    BestFour(int sr, String sn, int m[]) {
        super(sr, sn, m);
        tot = avg = 0.0;
    }
    void calculate() {
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 5 - i - 1; j++) {
                if (mark[j] < mark[j + 1]) {
                    int t = mark[j];
                    mark[j] = mark[j + 1];
                    mark[j + 1] = t;
                }
            }
        }
        tot = mark[0] + mark[1] + mark[2] + mark[3];
        avg = tot / 4.0;
    }
    void display() {
        super.display();
        System.out.println("Total of best of four " + tot);
        System.out.println("Average of best of four " + avg);
    }
}
```

12.
```
class Word {
    String str;
    int len;
    Word(String s) {
        str = s;
        len = str.length();
    }
    void display() {
```

```
            System.out.println("Original Word " + str);
        }
    }
    class ChangeVowel extends Word {
        String nstr;
        ChangeVowel(String s) {
            super(s);
            nstr = "";
        }
        void change() {
            String v = "AEIOU";
            int p = -1;
            str = str.toUpperCase();
            for (int i = 0; i < len; i++) {
                char c = str.charAt(i);
                if (v.indexOf(c) != -1) {
                    p = (v.indexOf(c) + 1) % 5;
                    nstr = nstr + v.charAt(p);
                } else {
                    nstr = nstr + c;
                }
            }
        }
        void display() {
            super.display();
            System.out.println("Changed Word " + nstr);
        }
    }
```

13. Data Structure

Unsolved Questions

A. 1. b 2. b 3. d 4. c 5. c
6. a 7. c 8. b 9. c

B. 1. non linear 2. peek 3. one 4. root 5. path
6. Binary Search Tree 7. one 8. post order 9. x y z ^ %
10. reverse polish

C. 1. a.

Array	Linked List
Contiguous memory allocation	Non Contiguous memory allocation
Fixed size	It can grow or shirnk ie dynamic size

b.

Post fix	Prefix
Operator comes after operands	Operator comes before operands
Technique store then change	Technique change then store

c.

Height of Binary tree	Depth of binary tree
The height of a node is the length of the longest path from that node to a leaf	The depth of a node is the number of edges from the root to that node
Height of leaf node is 1	Depth of leaf node is 1

d.

Linked list	Binary tree
A linear data structure	A non linear data structure
Traversal is sequential, starting from the head	Traversal can be pre-order, in-order, post-order

e.

Pre order traversal	Post Order traversal
Order of traversal is Root -> Left -> Right	Order of traversal is Left -> Right -> Root
Visits root first	Visits root last

2. a. X Y Z - + W E + F * J / +
b. A B C / D E ^ * F % +
c. A B * C * E / F ^ G +
d. P Q R * + S - T * U V + / H /
e. X Y Z * W E + F * % + J –
f. A B + C D E - * / F G * +
g. A B + C D E + * ^ F - G –
h. P Q + R - S + T * U V + / H *
i. P Q R ^ + S - T % U V - / H +
j. X Y Z * / W E F * + - J +

3. a. ++X*YZ/^+WEFJ
   b. +A%*/BC^DEF
   c. -+-^/**ABCEFGHI
   d. //*-+P*QRST+UVH
   e. +X*%*YZ+WE-FJ
   f. +/+AB*C-DE*FG
   g. --^+AB*C+DEFG
   h. */+++P*QRST+UVH
   i. +/%-+P^QRST-UVH
   j. +-/X*YZ+W*EFJ

4.
```
class Email
{      String msg[]=new String[50];
int size, top;
   Email(int x)
            {  size=x;
      top= -1;
      msg = new String[size];
}
void pushMail(String m)
{  if( top<size-1)
      msg[++top]=m;
   else
      System.out.println("Mail cannot be stored ... overflow");
}
void  popMail()
{  if(top>=0)
      System.out.println(msg[top--]+" popped ");
   else
      System.out.println("Empty ..... Underflow");
}
void display()
{  for(int i=top;i>=0;i--)
      System.out.println(msg[i]);
} }
```

5.
```
class CharStk {
    char ch[];
    int cap, top;
    CharStk(int max) {
        cap = max;
        top = -1;
        ch = new char[cap];
    }
    void pushChar(char c) {
        if (top < cap - 1)
            ch[++top] = c;
        else
            System.out.println("overflow");
    }
    char popChar() {
        if (top >= 0)
            return ch[top--];
        else
            return '\0';
    }
    void display() {
        for (int i = top; i >= 0; i--)
            System.out.println(ch[i]);
    }
}
```

6.
```
class Vehicle {
    String reg[] = new String[100];
    int cap, f, r;
    Vehicle(int v) {
        cap = v;
        f = r = 0;
        reg = new String[cap];
    }
    void addVehicle(String v) {
```

```
            if (r < cap - 1)
                reg[++r] = v;
            else
                System.out.println("Overflow");
        }
        String delVehicle() {
            if (f != r)
                return reg[++f];
            else
                return "#Empty#";
        }
        void disp() {
            for (int i = f + 1; i <= r; i++)
                System.out.println(reg[i]);
        }
    }
```

7.
```
class Stack {
        String st[];
        int size, top, ctr;
        Stack() { // Default constructor
            size = 10; // Default size (you can choose)
            top = -1;
            ctr = 0;
            st = new String[size];
        }
        Stack(int cap) { // Parameterised constructor
            size = cap;
            top = -1;
            ctr = 0;
            st = new String[size];
        }
        void pushname(String n) {
            if (top < size - 1) {
```

```
                st[++top] = n;
                ctr++;
            } else
                System.out.println("Overflow");
        }
        String popname() {
            if (top >= 0) {
                ctr--;
                return st[top--];
            } else
                return "underflow";
        }
        void display() {
            for (int i = top; i >= 0; i--)
                System.out.println(st[i]);
        }
    }
```

8.
```
    Class JobNo
    {      int Q[], size,start,end;
       JobNo(int max)
                {   size=max;
          start=end=0;
          Q= new int[size];
    }
    void enter(int  j)
    {   if( end<size-1)
          Q[++end]=j;
       else
          System.out.println("NO SPACE");
    }
    int remove()
    {   if(start !=end)
          return(Q[++start]);
       else
```

```
        return -1;
    }
    void show()
    {   for(int i=start+1;i<=end; i++)
        System.out.println(Q[i]);
    } }
```

9.
```
    class Link
    {       int lnk[], max,begin,end;
       Link(int nn)
                {   max=nn;
        begin=end=0;
        lnk= new int[max];
    }
    void addlink(int v)
    {   if( end<max-1)
        lnk[++end]=v;
       else
        System.out.println("OUT OF SIZE");
    }
    int dellink()
    {   if(begin !=end)
        return(lnk[++begin]);
       else
              { System.out.println("EMPTY...");
        return -99; }
    }
    void display()
    {   for(int i=begin+1;i<=end;i++)
        System.out.println(lnk[i]);
    }
    }
```

10.
```
    class BothEnd
    {       int ele[], cap, front,rear;
       BothEnd(int nn)
```

```
            {   cap=nn;
        front=rear=0;
        ele= new int[cap];
    }
    void addfront(int v)
    {   if(front !=0)
        ele[front--]=v;
       else
        System.out.println("full from front..overflow");
    }
    void addrear(int v)
    {   if( rear<cap-1)
        ele[++rear]=v;
       else
        System.out.println("full from rear..overflow");
    }
    int  delfront()
    {   if(front !=rear)
        return(ele[++front]);
       else
        return -999;
    }
    int delrear()
    {   if(front !=rear)
        return(ele[rear--]);
       else
        return -999;
    }
    void display()
    {   for(int i=front+1;i<=rear;i++)
        System.out.println(ele[i]);
     } }
```

11.
```
class Circle {
    int ele[], cap, front, rear;
```

```
    Circle(int max) {
        cap = max;
        front = rear = 0;
        ele = new int[cap];
    }
    void addvalue(int v) {
        if ((rear + 1) % cap != front) {
            rear = (rear + 1) % cap;
            ele[rear] = v;
        } else
            System.out.println("Queue full..overflow");
    }
    int delValue() {
        if (front != rear) {
            front = (front + 1) % cap;
            return ele[front];
        } else
            return -1;
    }
    void display() {
        int i = front;
        while (i != rear) {
            i = (i + 1) % cap;
            System.out.println(ele[i]);
        }
    }
}
```

12. Algorithm:

Step 1: Start.

Step 2: Create a new Link ptr and initialise it to start

Step 3 : Initialise high=ptr.n

Step 4: Repeat Step 5 to Step 6 while ptr != null

Step 5: if high < ptr.n then assign high=ptr.n

Step 6: Assign ptr = ptr.next

Step 6: return high

Step 7 : Stop

Java Code:

```
int FindHighest(Link start) {
    if (start == null) {
        return -1;
    }
    Link ptr = start;
    int high = ptr.n;
    while (ptr != null) {
        if (high < ptr.n)
            high = ptr.n;
        ptr = ptr.next;
    }
    return high;
}
```

13. Algorithm:

Step 1: Start.

Step 2: Create a new Link Nm ptr and initialise it to head

Step 3: Repeat Step 4 to Step 5 while ptr != null

Step 4: if ptr.name.charAt(0) is equal to 'A' then display ptr.name

Step 5: Assign ptr = ptr.next

Step 6: Stop

Java Code:

```
void printA(LinkNm head) {
    LinkNm ptr = head;
    while (ptr != null) {
        if (ptr.name.charAt(0) == 'A')
            System.out.println(ptr.name);
        ptr = ptr.next;
    }
}
```

14. Step 1: Start

Step 2: Create Node prev and initialise it to null

Step 3: Create Node current and initialise it to startptr

Step 4: Create Node next and initialise it to null

Step 5: Repeat Steps 6 to Step 9 while current!=null

Step 6: Assign next as current.link

Step 7: Assign current.link as prev

Step 8: Assign prev as current

Step 9: Assign current as next

Step 10: Assign startptr=prev

```
void  reverse(Node startptr) {
  Node prev = null;
  Node current = startptr;
  Node next = null;
  while (current != null) {
    next = current.link;
    current.link = prev;
    prev = current;
    current = next;
  }
  return prev;
}
```

15. a. A

b. H,I

c. Pre Order : D,F,J

In Order : F,D,J

Post Order : F,J,D

d. Node E Level : 2 Height : 3 Depth : 2

Node B Level : 1 Height : 4 Depth : 2

e. A,B,C,D,E,G

f. (B,C), (E,I), (F,J)

g. Parent Node : A Child Node : E,I

h. Size : 10 Degree : 2

i. Height of left subtree is 4 and right subtree is 5. Difference of their height is 1. So the tree is balanced.

j. It is not a complete binary tree as level 1 and level 2 are not completely filled.

14. Computational Complexity and Big O notation

Unsolved Questions

A. 1. c 2. c 3. d 4. a 5. b

B. 1. O(log n) 2. O(n) 3. Constants 4. $O(N^2)$ 5. Best

C. 1. a. Space and time complexity

Space complexity: It is the amount of memory space required to run an algorithm. An algorithm which requires the least memory is considered more efficient.

Time complexity: It is the total time taken to execute the instructions (steps) of a program. An algorithm which is designed using the least number of steps is considered more efficient. However, time complexity is a function of input size.

b. Best case and worst case complexity

Best case complexity: For any input of size 'n', the function takes the minimum time or minimum number of steps for execution is called the best case complexity. It defines the lower bound of an algorithm.

Worst case complexity: For any input of size 'n', the function takes the maximum time or the maximum number of steps required for execution is called the worst case complexity. It defines the upper bound of an algorithm.

c. Complexity $O(N^2)$ and O(log N)

$O(N^2)$: The change is quadratic in nature and is directly proportional to the square of the input size. It is less efficient than O(log n). Example bubble sort, selection sort, insertion sort.

O(log n): The change is logarithmic in nature. It is more efficient than $O(n^2)$. Example binary search.

2. a. O(N): The change in complexity is also linear and is directly proportional to the number of inputs. If the input size increases or decreases by 'x', the growth also changes by 'x'.

b. O(N log N): Log-linear time complexity.: The complexity is better than some other sorting algorithms where complexity changes in quadratic manner.

c. O(N!): The algorithm has a run time proportional to the factorial of the input size. When n=2, the algorithm runs 2 times, 6 times for n=3, 24 times for n=4 and so on.

d. O(1): Constant time complexity.The time taken is constant and does not grow with the input size. For example, push and pop operation in stack

e. $O(2^N)$: The algorithm doubles with every additional input. When n is 2, 4 steps are executed, which becomes 8 when n is 3 and 16 when n is 4.

3. Big 'O' notation is a mathematical notation used to describe the upper bound of an algorithm's time or space complexity in terms of the size of the input. It provides a way to express the worst-case scenario of how an algorithm's performance scales as the input size increases

   The two factors are :

   Space complexity: It is the amount of memory space required to run an algorithm. An algorithm which requires the least memory is considered more efficient.

   Time complexity: It is the total time taken to execute the instructions (steps) of a program. An algorithm which is designed using the least number of steps is considered more efficient. However, time complexity is a function of input size.

4. The complexity of the first loop is O(a)

   The complexity of the second loop is O(b.c)

   So the complexity is O(a + b.c)

   If it is replaced by N then the complexity will be $O(N+N^2)$. Considering the dominant term the complexity will be $O(N^2)$

5. a. if(top == size - 1):

      This is a simple comparison operation, which takes constant time, i.e., O(1)

      System.out.println(...):

      Printing a message to the console also takes constant time, O(1)

      a[++top] = n;:

      This involves incrementing top and assigning a value to an array element. Both operations are constant time, i.e., O(1)

      So the complexity is O(1)

   b. Node ptr = start;:

      This is a simple assignment operation, which takes constant time, O(1)

      start = ptr.next;:

      Accessing the next field of a node and assigning it to start also takes constant time, O(1)

      ptr.next = null;:

      Setting a pointer to null is another constant time operation, O(1)

      So the complexity is O(1)

   c. int i = 1;:

      This is a simple initialization, which takes constant time, O(1)

      do {...} while(i <= n);:

Starting from i = 1, the loop executes until i reaches n. This means the loop runs n times. So it's complexity is O(n)

Inside the do-while loop, the operations are:

Printing i with System.out.println(i) takes O(1) time.

Incrementing i with i = i + 1 also takes O(1)time.

The final complexity is O(n)

d. for(int i = 1; i < x; i++):

This loop runs from x-1 times, which simplifies to O(x)

The if inside loop and sum calculation has complexity of O(1) each

The if outside loop and print statements have complexity O(1)

So the complexity is O(n)