1. System of Numeration

MIND DRILL

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

B. 1. digit, base 2. 2 3. 0 to 9 4. A to F 5. repeated division by 2
6. 010010, 010011 7. 1560, 1561 8. FA 9. 1 10. 1110

C. 1. Decimal to Binary:

a. $(763)_{10} = (1011111011)_2$
b. $(2154)_{10} = (100001101010)_2$
c. $(678.50)_{10} = (1010100110.1)_2$
d. $(2351.75)_{10} = (100100101111.11)_2$

2. Decimal to Octal:

a. $(299)_{10} = (453)_8$
b. $(3846)_{10} = (7406)_8$
c. $(178.48)_{10} \approx (262.36)_8$
d. $(4367.5)_{10} = 10417.4_8$

3. Decimal to Hex:

a. $(488)_{10} = 1E8_{16}$
b. $(3926)_{10} = F56_{16}$
c. $(978.5)_{10} = 3D2.8_{16}$
d. $(6363.25)_{10} = 18DB.4_{16}$

4. Binary to Decimal:

a. $(100110)_2 = 38_{10}$
b. $(11101111)_2 = 239_{10}$
c. $(101010.11)_2 = 42.75_{10}$
d. $(1111010.101)_2 = 122.625_{10}$

5. Octal to Decimal:

a. $(2571)_8 = 1401_{10}$
b. $(3773)8 = 2043_{10}$
c. $(1655.25)_8 = 941.328125_{10}$
d. $(2511.4)8 = 1353.510$

6. Hex to Decimal:
   a. $(2DC1)_{16} = 11713_{10}$
   b. $(EC98)_{16} = 60568_{10}$
   c. $(5E67.8)_{16} = 24167.5_{10}$
   d. $(E22B.C)_{16} = 57899.75_{10}$
7. Binary to Octal:
   a. $(10011110)_2 = (236)_8$
   b. $(111011101)_2 = (735)_8$
   c. $(1001010.11)_2 = (112.6)_8$
   d. $(1110010.001)_2 = (162.1)_8$
8. Binary to Hexadecimal:
   a. $(100111100)_2 = (13C)_{16}$
   b. $(11101110011)_2 = (773)_{16}$
   c. $(10010101.101)_2 = (95.A)_{16}$
   d. $(11110010.001)_2 = (F2.2)_{16}$
9. Octal to Binary:
   a. $(4573)_8 = (100101111011)_2$
   b. $(1771)_8 = (1111111001)_2$
   c. $(655.25)_8 = (110101101.010101)_2$
   d. $(536.4)_8 = (101011110.100)_2$
10. Hexadecimal to Binary:
   a. $(DEC)_{16} = (110111101100)_2$
   b. $(FF98)_{16} = (1111111110011000)_2$
   c. $(5E.8)_{16} = (1011110.1000)_2$
   d. $(D2B.C)_{16} = (110100101011.1100)_2$
11. Binary Addition
   a. $(101010)_2 + (1101)_2 = (110111)_2$
   b. $(111101010)_2 + (110110)_2 = (1000100000)_2$
   c. $(110111.01)_2 + (111.11)_2 = (111111.00)_2$
   d. $(1101101.11)_2 + (101.11)_2 = (1110011.10)_2$
12. Octal Addition
   a. $(354)_8 + (205)_8 = (561)_8$
   b. $(1266)_8 + (505)_8 = (1773)_8$
   c. $(1057.6)_8 + (2341.5)_8 = (3421.3)_8$
   d. $(344.25)_8 + (65.25)_8 = (431.5)_8$
13. Hexadecimal Addition
   a. $(A67)_{16} + (2175)_{16} = (2BDC)_{16}$
   b. $(B23C)_{16} + (D16)_{16} = (BF52)_{16}$
   c. $(A9.CD)_{16} + (5.EE)_{16} = (AF.BB)_{16}$
   d. $(678.9)_{16} + (ABC.D)_{16} = (1135.6)_{16}$
14. Binary Subtraction
   a. $(101010)_2 - (1101)_2$
      (i) Borrow method
      $= (011101)_2$
      (ii) One's complement method
      $= (11101)_2$
      (iii) Two's complement method
      $= (11101)_2$

b. $(111101010)_2$ - $(110110)_2$

(i) Borrow method

= $(111010100)_2$

(ii) One's complement method

= $(111010100)_2$

(iii) Two's complement method

= $(111010100)_2$

c. $(110111.01)_2$ - $(111.11)_2$

(i) Borrow method

= $(101111.10)_2$

(ii) One's complement method

= $(101111.10)_2$

(iii) Two's complement method

= $(101111.10)_2$

d. $(1101101.11)_2$ - $(1001.01)_2$

(i) Borrow method

= $(1100100.10)_2$

(ii) One's complement method

= $(1100100.10)_2$

(iii) Two's complement method

= $(1100100.10)_2$

15. Octal Subtraction

a. $(4541)_8$ - $(205)_8$

(i) Borrow method

= $(4334)_8$

(ii) One's complement method

= $(4334)_8$

(iii) Two's complement method

= $(4334)_8$

b. $(12.66)_8$ - $(5.75)_8$

(i) Borrow method

= $(4.71)_8$

(ii) One's complement method

= $(4.71)_8$

(iii) Two's complement method

= $(4.71)_8$

c. $(1057.6)_8$ - $(2341.5)_8$

(i) Borrow method

= $-(1261.7)_8$

(ii) One's complement method

= $-(1261.7)_8$

(iii) Two's complement method

= $-(1261.7)_8$

d. $(3445)_8$ - $(5625)_8$

(i) Borrow method

= $-(2160)_8$

(ii) One's complement method

= $-(2160)_8$

(iii) Two's complement method

= $-(2160)_8$

16. Hexadecimal Subtraction

a. $(B78)_{16}$ - $(8AB)_{16}$

(i) Borrow method

= $(2CD)_{16}$

(ii) One's complement method

= $(2CD)_{16}$

(iii) Two's complement method

= $(2CD)_{16}$

b. $(B23C)_{16}$ - $(DF6)_{16}$

(i) Borrow method

= $(A446)_{16}$

(ii) One's complement method

= $(A446)_{16}$

(iii) Two's complement method

= $(A446)_{16}$

c. $(A9.CD)_{16}$ - $(5.EE)_{16}$

(i) Borrow method

= $(53.DF)_{16}$

(ii) One's complement method

= $(53.DF)_{16}$

(iii) Two's complement method

= $(53.DF)_{16}$

d. $(678.9)_{16}$ - $(AFC.D)_{16}$

(i) Borrow method

= $(-2E3.1)_{16}$

(ii) One's complement method

= $(-2E3.1)_{16}$

(iii) Two's complement method

= $(-2E3.1)_{16}$

17. Octal Multiplication

a. $(342)_8 \times (233)_8 = 104326_8$

b. $(2642)_8 \times (212)_8 = 604524_8$

c. $(3664)_8 \times (1531)_8 = 6344624_8$

d. $(4172)_8 \times (45)_8 = 234642_8$

18. Hexadecimal Multiplication

a. $(75A)_{16} \times (219)_{16} = F6BCA_{16}$

b. $(46D)_{16} \times (39)_{16} = FC45_{16}$

c. $(CAB)_{16} \times (44)_{16} = 35D6C_{16}$

d. $(EF)_{16} \times (611)_{16} = 5A9DF_{16}$

D. 1. Carry occurs when sum exceeds 1. It is forwarded to the next left bit. This ensures correct positional value calculation in binary arithmetic used in digital circuits.

2. Binary is used for internal processing, octal and hexadecimal are used for compact representation. They simplify programming, memory addressing, debugging, and reduce complexity of long binary numbers.

E. 1. Different score values for players in a game

2. Decimal scores: 72, 108, 92, 115

3. Final total (decimal): 387

4. Final binary sum: 100110001_2

2. Encodings

MIND DRILL

A. 1. b. 2. c. 3. d. 4. b. 5. b. 6. d.

B. 1. 48 to 57 2. ISCII 3. Unicode Transformation Format
4. byte 5. Unicode 6. gigabytes

C. 1. UTF-16 uses 2 bytes to represent 65536 characters and 4 bytes for the additional characters. It is used in operating systems like Microsoft Windows. UTF-32 represents each character using 4 bytes.

2. ASCII is the most popular coding scheme used as industry standard in computers and on the web. It was introduced in 1960. Originally, ASCII was designed as a 7-bit character set having 27 or 128 characters. It could represent all digits from 0 to 9, uppercase and lowercase English alphabets and some special characters.

   a.Uppercase letters: 65 to 90 b. Lowercase letters: 97 to 122

3. To represent alphabets in BCD, a 6-bit coding system was developed which could represent numbers, letters and some special characters. It contains 2 zone bits and 4 numeric bits and can be represented using octal equivalents. Internally, however, it appears as a 7-bit code, the leftmost bit called the check bit or parity bit.

4. A character set consists of a group of characters that are used to represent a particular language. The character set of the English language is different from that of Hindi, Bengali, Chinese or any other languages spoken in different parts of the world. The visual representation of a character is called a glyph. To represent these character sets, the computer uses the concept of a coded character set, where each character is assigned a unique number called code points. The code points are commonly referred to in decimal notation, though hexadecimal notation can also be used.

5. ISCII uses attribute codes (ATR) followed by byte values to apply formatting like bold, italic, underline and to switch between Indian scripts such as Hindi, Tamil, Bengali and others.

6. Unicode supports all world languages, scripts and symbols, ensures compatibility with modern technologies like HTML, Java and eliminates encoding conflicts between different systems.

7. Limitations of BCD

   i. BCD codes are inefficient as compared to binary codes. For example, 12 in binary is 1100 but in BCD notation it is written as 0001 0010.

ii. Arithmetic operations become more complex as compared to binary notation.

iii. As BCD is a 4-bit code, it can only represent numbers and is incapable of handling non-numeric characters.

8. Each decimal digit is converted separately into its 4-bit binary equivalent and then combined. This allows correct representation but makes processing slower compared to binary.

D. 1. Carry occurs when sum exceeds 1. It is forwarded to the next left bit. This ensures correct positional value calculation in binary arithmetic used in digital circuits.

2. Binary is used for internal processing, octal and hexadecimal are used for compact representation. They simplify programming, memory addressing, debugging, and reduce complexity of long binary numbers.

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

3. Propositional Logic, Hardware Implementation, Arithmetic Operations

MIND DRILL

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

B. 1. A′.B′ + A.B 2. OR gate 3. A ⊕ B 4. Fundamental
5. Odd 6. XOR 7. NAND 8. All 9. Converse
10. $A.B + B.C_{in} + A.C_{in}$

C. 1. a.

Simple proposition	Compound proposition
It contains a single atomic statement.	It contains two or more simple propositions joined by special symbols called connectives.

b.

Conjunction (AND)	Disjunction (OR)
It is a binary connective that results in true only if all propositional variables are true.	It is a binary connective that results in true if any one proposition is true.
It is represented by . or Λ.	It is represented by + or $\vee$.

c.

Conditional (Implication)	Biconditional (Equivalence)
It is represented by $\rightarrow$.	It is represented by $\leftrightarrow$.
It is false only when the first proposition is true and the second is false.	It is true when both propositions have the same truth value and false when they have different truth values.

d.

AND gate	OR gate
Produces high output (1) only when all the inputs are high (1).	Produces high output (1) when any one input is high (1).
AND operation is represented by a dot (.).	OR operation is represented by a plus (+).

e.

Half adder	Full adder
Adds two binary bits.	Adds three binary bits (including carry input).
Has two inputs and two outputs (Sum and Carry).	Has three inputs and two outputs (Sum and Carry).
Does not accept a carry input.	Accepts a carry input from the previous stage.

2. a. $Q \rightarrow (P \wedge R)$ b. $P \wedge Q$ c. $Q \leftrightarrow P$
   d. $\sim P \vee \sim Q$ e. $Q \rightarrow R$
3. a. It is not raining heavily or there is no prediction for a cyclone.
   b. It is not raining heavily or there is no prediction for a cyclone and there will be no damage to life and property.
   c. It is not raining heavily if and only if there is no prediction for a cyclone.
   d. It is not true that it is raining heavily and there is a prediction for a cyclone and there will be damage to life and property.
   e. If there is no prediction for a cyclone, then there will be no damage to life and property.
4. a. **Converse:** If you crack IIT, then you work hard.

   Inverse: If you do not work hard, then you will not crack IIT.

   Contrapositive: If you do not crack IIT, then you do not work hard.

   b. **Converse:** If it is a rose, then the flower has a beautiful smell.

   Inverse: If the flower does not have a beautiful smell, then it is not a rose.

   Contrapositive: If it is not a rose, then the flower does not have a beautiful smell.

   c. **Converse:** If I 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 do not wear my red gown, then I am not invited to a party.

d. **Converse:** If we celebrate, then India wins World Cup cricket.

Inverse: If India does not win World Cup cricket, then we will not celebrate.

Contrapositive: If we do not celebrate, then India does not win World Cup cricket.

e. **Converse:** If it is a palindrome, then the number is equal to its reverse.

Inverse: If a 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. Contingency b. Contingency c. Contingency
   d. Contingency e. Tautology f. Tautology
   g. Contingency h Tautology i. Contradiction
   j. Tautology

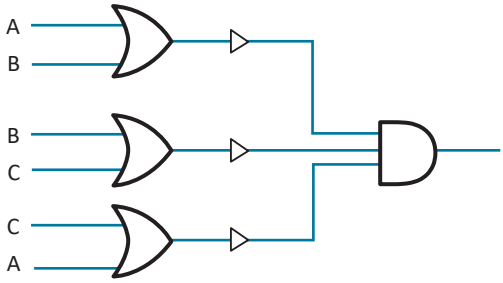
6.

P	Q	$P \rightarrow Q$	$(P \rightarrow Q) \rightarrow P'$	$\sim[(P \rightarrow Q) \rightarrow P']$
0	0	1	1	0
0	1	1	1	0
1	0	0	1	0
1	1	1	0	1

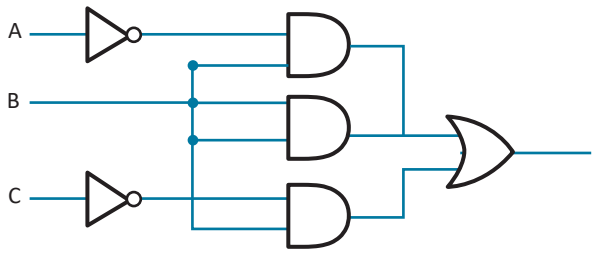
7. a. A.B b. A.C c. (A.B + (A.C)')'

8. a. A.B b. A.C c. (A.B + (A.C)')'

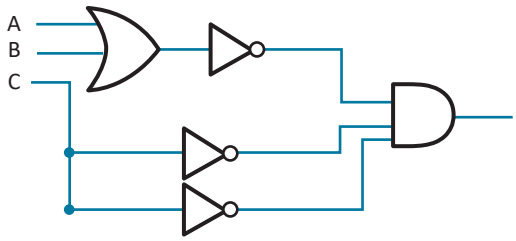
9. a.



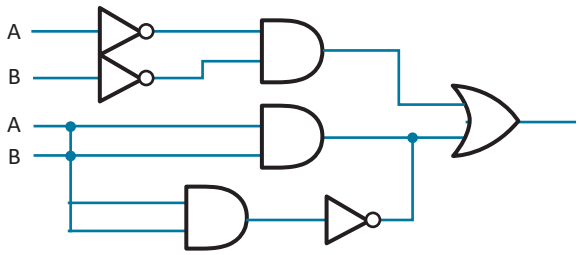
b.



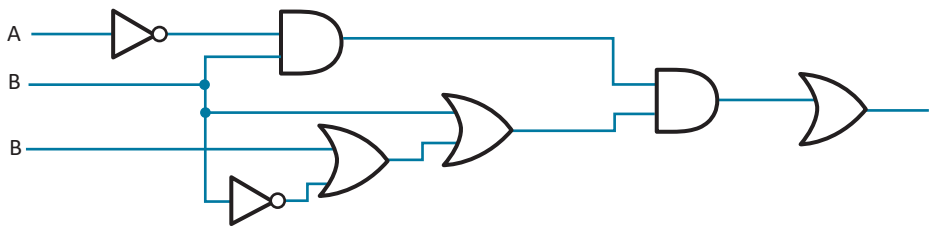
c.



d.



e.



D. 1. A NOR gate outputs a HIGH (1) signal only when all its inputs are LOW (0).

- **Setup:** Design the system using active-low sensors (where a secure state outputs 1 and a triggered/broken state outputs 0). Connect the window, door, and motion sensors to the inputs of a NOR gate.
- **Result:** When all sensors are intact (1, 1, 1), the alarm output is 0 (OFF). If the specific combination of monitored sensors is breached and all drop to 0 (0, 0, 0), the NOR gate outputs a 1, activating the alarm.

2. **Expression:** (A Λ B) $\vee$ (A$\Lambda \sim$ B)

   Simplification:

   1. Distributive Law: A . (B + B′)
   2. Complement Law (B + B′ = 1): A . 1

3. **Identity Law:** A

   Simplified Form: A

   What it represents: This represents the Law of Simplification. It shows that the variable B is entirely redundant; the output depends solely on the state of variable A. In a real circuit, gates tracking B can be completely removed to save cost and space.

E. 1. Boolean Expression for Motor M

The conditions for the motor M to run are combined using logical connectives:

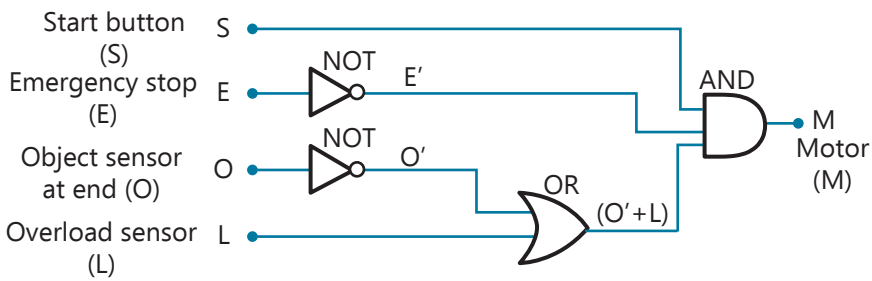
- **Condition 1:** The Start button must be pressed
- **Condition 2:** The Emergency stop button must not be pressed
- **Condition 3:** Either there should be no object at the end OR an overload detected

Since all three primary conditions must be satisfied simultaneously, they are combined using a conjunction (AND) operation: M = S . E′ . (O′ + L)

2.

S	E	O	L	E′	O′	(O′+L)	Motor (M)
0	0	0	0	1	1	1	0
0	0	0	1	1	1	1	0
0	0	1	0	1	0	0	0
0	0	1	1	1	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	0
0	1	1	0	0	0	0	0
0	1	1	1	0	0	1	0
1	0	0	0	1	1	1	1
1	0	0	1	1	1	1	1
1	0	1	0	1	0	0	0
1	0	1	1	1	0	1	1
1	1	0	0	0	1	1	0
1	1	0	1	0	1	1	0
1	1	1	0	0	0	0	0
1	1	1	1	0	0	1	0

3.



Boolean Expression:
M = S * E′ * (O′ + L)

S	=	Start button
E	=	Emergency stop (1 when pressed)
O	=	Object sensor at end (1 if object detected)
L	=	Overload sensor (1 if overload detected)
M	=	Motor output (1 when motor runs)

4. Introduction to Object-Oriented Programming Using Java

MIND DRILL

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

B. 1. object-oriented 2. main() 3. runtime 4. abstract
5. many forms

C. 1. Some of the characteristics of object-oriented programming language are as follows:

- It follows a bottom-up approach.
- More emphasis is given on data rather than procedure.
- Programs are divided into objects.
- Data cannot be accessed by external functions.
- Functions are the ways used by objects to communicate with each other.
- The most important characteristic is that new data and functions can be added as and when required without making severe changes to the program. This is because it uses the concept of reusability where coding is written once and can be used multiple times using different objects.

2. Java applets are the small Java programs that are implanted in an HTML page and execute only within any web browser. They are mainly used for executing internet programming where graphics, sounds and accepting user input are very important. It executes on the client side. Whereas Java standalone applications can run independently on an individual computer. In Java, the main() method is the first function that starts the execution of every standalone application.

3. Java was developed by Sun Microsystems in 1991 by James Gosling and Patrick Naughton. It was initially called OAK and later renamed Java in 1995. It was designed for embedded systems but later became widely used for internet programming, gaming and mobile applications due to its platform independence and object-oriented nature.

Java used the principle of WORA, i.e., Write Once, Run Anywhere. Thus, it provides free execution on any platform such as Windows, Linux, etc. Also, it has the properties of configurable security, file access restrictions, etc. As it can run on the network, major web browsers incorporated the ability to run Java applets within web pages. This made Java very popular.

4. The main features of BlueJ are as follows:
   - **BlueJ has a simpler and smaller interface than other specialised environments like NetBeans and Eclipse.**
   - **Since BlueJ is a Windows-based environment, it is very easy to interact with objects.**
   - **BlueJ can run on Windows, (macOS X), Linux and other platforms.**
   - **BlueJ provides a menu-driven window approach.**
   - **Projects from non-BlueJ environments can be exported and imported.**
5. There are some errors that will execute the program but the desired output is not achieved. This type of error is called a logical error.

   Errors in program:
   - **s should be initialised to 0 (not 1)**
   - **Loop should be i <= 10 instead of i < 10**

   Correct logic:
   - **Correct sum calculation requires proper initialisation and correct loop condition.**
6. The procedure of creating a new class with the help of a class already created by using its properties and functionality is said to be inheritance.

   Example: A Car class inherits from Motor Vehicle, which inherits from Vehicle, reducing code duplication and improving reusability.
7. Some of the drawbacks of procedure-oriented programming language are as follows:
   - Data is exposed to the whole program which makes the security of the program under pressure.
   - Reusability of code is not allowed.
   - It is impossible to relate to real-life objects.
   - It is difficult to create a new data type.
8. Some more features of the Java language are as follows:
   - It uses both the compiler as well as the interpreter.
   - Java is considered to be more dynamic than C and C++ as it is designed to adjust to a progressing environment.
   - It is a case-sensitive language.

- It enables high performance.
- It tries to eliminate error-prone situations as it stresses compile-time error and runtime scrutiny.

D. 1. Inheritance helps by creating a common base class Vehicle that contains shared properties like speed, fuel type, and mileage.

The derived classes Car and Bike reuse these properties instead of rewriting them. This reduces code duplication and improves reusability because common code is written once in the parent class and inherited by all child classes.

2. Java supports cross-platform execution using the Write Once, Run Anywhere (WORA) principle.

The Java compiler converts code into bytecode, which is then executed by the Java Virtual Machine (JVM). Since JVM is available for Windows, Linux and other operating systems, the same program runs on different platforms without modification.

E. 1. b. 2. c. 3. c. 4. c.

5. Objects

MIND DRILL

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

B. 1. objects 2. runtime exceptions 3. object 4. data
5. Java bytecode

C. 1. Some features of JVM are as follows:

- It allows Java code to execute on any computers, mobiles, etc. Also, it can run on any operating system. It follows the principle of "Write Once, Run Anywhere".
- It manages and optimises the program so that memory usage can be minimised.

2. When Java source code is compiled by the Java compiler (JAVAC), the resultant code is called Java bytecode, which is further converted by the Java virtual machine to machine-dependent code.

3. To use a Scanner class, follow the steps given below:

Step 1: Import the package at the beginning of the program. There are two ways to import as:

1. import java.util.*;
2. import java.util.Scanner;

Step 2: Then we have to create the object of the Scanner class. The syntax to create the object is as follows:

Scanner sc = new Scanner(System.in);

where sc is the name of the object of the Scanner class.

Step 3: Use Scanner class methods as per your data types to read input from the user.

Say, to read the whole number from the user, we have to use "sc.nextInt();".

Similarly, to read a character from the user, we have to use "sc.next().charAt(0);".

4. A method can throw many types of exceptions during the execution of the program. The throws keyword declares all these types of exceptions. This is used to help the programmers with prior knowledge about what types of exceptions are to be handled.
5. Input/Output (I/O) exceptions are Java IOExceptions that occur whenever an input or output operation is unsuccessful. For example, while trying to read data from a file that is not present, Java throws an I/O exception named "FileNotFoundException".

   Some of the other exceptions are as follows:

   - ArithmeticException: If there is any error in the arithmetic operation then this error is thrown.
   - ArrayIndexOutOfBoundsException: It is thrown to indicate that an array has been accessed with an illegal index.
   - ClassNotFoundException: When we try to access a class, which is not present then this error is thrown.
   - NoSuchMethodException: When we try to access a method, which is not found then this error is thrown.
   - NullPointerException: This exception is thrown when referring to the members of a null object. Null represents nothing.
   - InputMismatchException: This exception is thrown when we try to enter one type of data to another which is not there.
6. The differences between class and object are given in the table and also shown in the picture below:

Class	Object
A class is a blueprint to create similar kinds of objects.	An object is a unique entity.
No memory is occupied when a class is created.	As soon as the object is created, memory is initialised.
Class is a user-defined data type.	Object is a variable of the user-defined data type known as class.
Class is the logical representation of data.	Object is the physical representation of data.
Class is created once in a program.	Object can be created many times as and whenever required just by utilising the name of the class.
Syntax for declaring a class: class <classname> { }	Syntax for instantiating an object for a class: <classname> <objectname> = new <constructor>;

7. It is used to handle exceptions and prevent program termination during runtime errors.
8. Class and object have the following properties:
    - Class is an object factory: Factories are the places that produce products of the same kind. A class acts as a factory as by using it, similar types of objects are created with different characteristics and common behaviours. We can also say a class is a blueprint of objects. Hence, a class is called an object factory.
    - Object is an instance of a class: As soon as an object is created in Java, it acquires memory in RAM, but this does not happen while defining a class. Thus, objects are the physical existence of the class.

      We know data types such as int, long, float, etc. are predefined classes in Java. Similarly, user-defined classes create objects that contain all the properties and methods of those classes and also acquire memory.

      Thus, we can say an object is an instance of a class.
    - Class is a user-defined data type: A user-defined data type is a derived data type depending on some existing data types.

D. 1. In this banking application, the NoSuchMethodException should be handled using proper exception handling so that the program does not crash when a user tries to call a non-existent method.

The method invocation should be placed inside a try-catch block, and the NoSuchMethodException should be caught separately. When this exception occurs, the system should display a user-friendly message such as "Invalid operation selected. Please choose a valid banking service." Instead of stopping the program, it should return the user to the main menu or available options so they can try again. This ensures the application remains stable, continues running smoothly, and provides a better user experience even when an invalid method is requested.

2. Java bytecode makes the application platform independent because it is not written for any specific operating system or hardware. After compilation, the Java compiler (JAVAC) converts the source code into bytecode, which is a universal intermediate code.

   This bytecode is executed by the Java Virtual Machine (JVM). Each device, whether it is a mobile phone, desktop, or server, has its own JVM implementation. The JVM interprets the same bytecode into machine-specific instructions.

   So, the same compiled bytecode can run on any system that has a JVM, without needing to be recompiled. This is why Java applications, like the financial app, are platform independent.

E. 1. b 2. a 3. c.

F. 1. Compound Interest Program

```
import java.util.*;


class compound_interest
{
    public static void main(String[] args)
```

```
    {
        Scanner sc = new Scanner(System.in);

        double p, r, t, ci;

        System.out.print("Enter the principal amount: ");
        p = sc.nextDouble();

        System.out.print("Enter the rate of interest: ");
        r = sc.nextDouble();

        System.out.print("Enter the time (in years): ");
        t = sc.nextDouble();

        ci = p * Math.pow((1 + r/100), t) - p;

        System.out.println("The compound interest is: " + ci);
    }
}
```

2. Multiplication of two integers

```
import java.util.*;

class multiplication
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);

        int a, b, m;

        System.out.print("Enter the first integer: ");
        a = sc.nextInt();

        System.out.print("Enter the second integer: ");
        b = sc.nextInt();
```

```
            m = a * b;

            System.out.println("The multiplication of " + a + " and "
    + b + " is: " + m);
        }
    }
```

3. Area of Circle

```
import java.util.*;

class circle_area
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        double r, area;

        System.out.print("Enter the radius of the circle: ");
        r = sc.nextDouble();

        area = 3.14 * r * r;

        System.out.println("Radius: " + r);
        System.out.println("Area of the circle: " + area);
    }
}
```

G. 1. Area of the triangle: 10

2. Enter the radius of the circle: 7
   Radius: 7.0
   Circumference of the circle: 43.98 (approx)
3. Enter the length of the rectangle: 5
   Enter the breadth of the rectangle: 8
   Length: 5.0
   Breadth: 8.0
   Diagonal of the rectangle: 9.433981132056603

6. Primitive Values, Wrapper Classes, Types and Casting

MIND DRILL

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

B. 1. Boolean 2. dynamic initialisation 3. Parentheses ()
4. data type 5. classes

C. 1. In implicit type conversion, the resulting data type is automatically determined by the compiler. This process is also known as widening conversion. An important point to note is that no loss of data occurs during implicit conversion. From the widening order, we can observe that a byte value can be converted to short, which can further be converted to int, long, float or double as required. Similarly, the char data type can be promoted to int, long, float or double.

2. Operators are the special symbols that instruct the compiler to perform some specific mathematical or non-mathematical operations on one or more operands. Java supports eight types of operators which are as follows:
   - Arithmetic Operators: "+", "-", "*", "/"
   - Assignment Operators: "="
   - Logical Operators: "&&", "||", "!"
   - Relational Operators: "==", "<", ">", "<=", ">=", "!="
   - Unary Operators: "+", "-", "++", "--"
   - Bitwise Operators: "&", "|", "^", "~"
   - Ternary Operators: "?:"
   - Shift Operators: (>>), (>>>), (<<)

3. \\ is used to insert a backslash character.

   \0 assigns a null value to a String variable.

4. An interface is an abstract class contains a combination of methods and variables. But the methods declared in an interface have signatures only. By default, all methods in an interface are abstract and the variables are public, static and final.

5. Size of long and byte data type in bits:
   - long = 64 bits
   - byte = 8 bits

6. An escape sequence is a non-graphical character preceded by a backslash (\) having a special meaning for the compiler. It is a command from the keyboard. Some of the escape sequences used in Java are as follows:

Escape Sequence	Required for
\t	Inserts a horizontal tab.
\b	Inserts a backspace.
\n	Inserts a new line.
\r	Inserts a carriage return.
\f	Inserts a form feed.
\'	Inserts a single quote character.
\"	Inserts a double quote character.
\\	Inserts a backslash character.
\0	Assigns a null to a String variable.
\v	Inserts a vertical tab.

Examples: \n new line, \t tab, \" double quote.

7. Implicit type conversion (also called type promotion) happens automatically in Java when a smaller data type is assigned to a larger data type. Since a long has a larger storage capacity than an int, Java automatically converts the int value into a long without any explicit instruction from the programmer. This process is safe because there is no risk of data loss when moving from a smaller type to a larger type.

   For example:

   ```
   int a = 100;
   long b = a;
   System.out.println(b);
   ```

8. Non-primitive data types are defined by the user. But they are built on primitive data types, i.e., they basically concatenate two or more same or different types of primitive data types. The different types of user-defined data types are String, class, array and interfaces. They are also known as reference data types, composite data types or user-defined data types.

   There are four types of Non-Primitive data type:

   • Class • Array • Interface • String

   Now, we will discuss the non-primitive data types in detail.

   Class: A class is a blueprint or template used to create objects. It defines the data members and methods that will be shared by all objects created from that class. For example,

   ```
   class sum
   {
   // data members
   // member methods
   }
   }
   ```

Array: Arrays can be defined as a collection of elements of the same data type stored under a single variable name. Each element in the array is identified by its index (subscript) which helps to access or separate the values. When we declare an array we create a structure that can store multiple values of the same type together.

The syntax to declare an array is as follows:

Primitive_Datatype variable_name[]= new Primitive_Datatype[size];

For example,

int ar[]= new int[10]; //it creates an integer array of 10 elements

Interface: An interface is an abstract class contains a combination of methods and variables. But the methods declared in an interface have signatures only. By default, all methods in an interface are abstract and the variables are public, static and final. For example,

```
interface fan
{
void show();
}
```

String: A string is a sequence of characters stored in a single variable. They are written within " " (double quotation marks). Unlike C/C++, Java strings are not terminated with a null character. Syntax of declaring a string:

String <variable> = "sequence_of_characters";

9. Wrapper classes are in-built classes that contain primitive data types. These classes are used to convert primitive data types into objects and vice-versa with the help of their methods. Every wrapper class has various methods. Thus, it provides a way to use the primitive data types (short, double, boolean, etc.) as objects. Every primitive data type is attached to its wrapper class. For example, the float data type is attached to the Float class, the double data type is attached to the Double class and so on.

```
class wrapper_class
{
   public static void main(String[] args)
   {
      Integer int_var = 100;
      Double double_var = 125.95;
      Character char_var = 'a';
      Float float_var = 34.56f;
      System.out.println("Integer Variable: " +int_var);
      System.out.println("Double Variable: " +double_var);
      System.out.println("Character Variable: " +char_var);
      System.out.println("Float Variable: " +float_var);
   }
}
```

10. The concept of explicit data type conversion is used to assign a value of a larger data type to a smaller data type. This process is known as narrowing. It is the reverse state of implicit data conversion. It requires the user's intervention.

    Before assigning the incompatible data type, we need to write down the data type within the brackets.

    Example 1:

    double d = 4.56;

    int n = (int)d;

    System.out.println("Result :" + n);

    Output: Result : 4

D. 1. When a program calculates the final price using both integer and floating-point values, implicit type conversion (type promotion) automatically occurs. In this case, the price of the item is given as an integer (₹1500), and the discount is given as a floating-point value (10.5%). During the calculation, the integer value is temporarily converted into a floating-point value by the system so that both values can be processed in the same data type. This ensures accurate calculation of the discount and final price.

    For example:

    Price = 1500 (int)

    Discount = 10.5% (float)

    1500 is converted to 1500.0 internally

    Discount amount = 1500.0 × 10.5 / 100 = 157.5

    Final price = 1500.0 − 157.5 = 1342.5

    However, mixing data types can sometimes lead to issues such as:

    - Loss of precision due to floating-point representation.
    - Rounding errors in financial calculations.
    - Unexpected results if implicit conversion is not properly understood.

    To handle these issues, programmers can:

    - Use consistent data types (preferably float or double for money calculations).
    - Apply explicit type casting where necessary.
    - Round off results to required decimal places for accuracy.
    - Use integer representation (like paise instead of rupees) in critical financial systems.

    Thus, implicit type conversion helps in smooth calculation, but careful handling is required to ensure accuracy.

2. If a product is added whose weight exceeds the maximum limit of the long data type, an overflow error may occur. This means the value cannot be stored correctly in the memory allocated for the long type. As a result, the stored value may become incorrect or wrap around to an unexpected value, leading to wrong calculations in the total weight of products.

Since long is a fixed-size data type, it can store values only within a specific range. When this range is exceeded, the program does not stop but produces incorrect results.

To handle this situation, type casting can be used. In type casting, the value can be converted into a higher data type such as double or a larger integer type before performing calculations. This helps in handling larger values more safely and reduces the risk of overflow during computation.

However, type casting alone does not increase the storage capacity of the original data type. Therefore, it is better to use a suitable higher-capacity data type from the beginning if large values are expected.

Thus, proper use of data types and type casting helps in avoiding errors and ensures accurate calculation of total weight.

E. 1. b. 2. c. 3. a. 4. a.

F. 1.

```
class Division
{
    public static void main(String[] args)
    {
        int a = 9, b = 3;
        float result = (float)a / b;
        System.out.println("Division Result: " + result);
    }
}
```

2.

```
class WrapperDemo
{
    public static void main(String[] args)
    {
        String i = "100”;
        String d = "10.5”;

        int a = Integer.parseInt(i);
        double b = Double.parseDouble(d);

        System.out.println("Converted Integer: " + a);
        System.out.println("Converted Double: " + b);
    }
}
```

3.
```
import java.util.*;
class Average
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        int a = sc.nextInt();
        int b = sc.nextInt();

        double avg = (a + b) / 2.0;
        System.out.println("Average : " + avg);
    }
}
```

G. 1. DiscountCalculator
Enter the original price: 200
Enter the discount percentage: 5
Discount Amount: 10.0
Final Price: 190.0

2. ImplicitConversion
Integer value: 25
Converted to double: 25.0
Character: A
ASCII value of character: 65
Integer converted to float: 25.0

3. Arithmetic program
Enter first number: 5
Enter second number: 7
Sum: 12
Difference: -2
Product: 35
Division (with precision): 0.7142857142857143

7. Variables and Expressions

MIND DRILL

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

B. 1. Relational operators 2. ternary operator
3. nested conditional operator 4. Equal to (==)
5. compound assignment operators
6. P = (a / (b * b)) + (b / (a * a))
7. Binary 8. 242 9. True 10. Bitwise XOR(^)

C. 1.

Arithmetic Expression	**Arithmetic Statement**
Any meaningful statement containing identifiers, literals and arithmetical operators which can produce a result is called an arithmetical expression.	When an arithmetical expression is assigned to a variable, then it is called an arithmetical statement.
Examples: a * b, 2 * (l + b)	Examples: c = a * b, p = 2 * (l + b)

2. Prefix vs postfix operators

Prefix increment/decrement operator: The prefix increment/decrement operator works on the principle of 'Change Before Action'. Given below is the example of the prefix increment operator. Here, an increment by 1 is executed first and then the value is used.

```
int a=5, b;
System.out.println("a: " +a);
b=++a;
System.out.println("a: " +a+ " and b: " +b);
```

Output: a: 5

a: 6 and b: 6

Given below is the example of the prefix decrement operator. Here, the decrement by 1 is executed first and then the value is used.

```
int a=5, b;
System.out.println("a: " +a);
b=--a;
System.out.println("a: " +a+ " and b: " +b);
```

Output: a: 5

a: 4 and b: 4

Postfix increment/decrement operator: The postfix increment/decrement operator works on the principle of 'Change After Action'. Given below is an example of the postfix increment operator. Here, the value is used first and then an increment by 1 is done on the given value.

```
int a=5, b;
System.out.println("a: " +a);
b=a++;
System.out.println("a: " +a+ " and b: " +b);
```

Output: a: 5

a: 6 and b: 5

Given below is an example of the postfix decrement operator. Here, the value is used first and then a decrement by 1 is done on the given value.

```
int a = 5, b;
System.out.println("a: " +a);
b = a--;
System.out.println("a: " +a+ " and b: " +b);
```

Output: a: 5

a: 4 and b: 5

3. a. Bitwise AND (&): When both the results are true, this operator returns true.

1st Bit	2nd Bit	1st Bit & 2nd Bit
0	0	0
0	1	0
1	0	0
1	1	1

b. Bitwise OR (|): When both the results are false, this operator returns false else it returns true.

1st Bit	2nd Bit	1st Bit | 2nd Bit
0	0	0
0	1	1
1	0	1
1	1	1

c. Bitwise Not(~): Bitwise Not or Bitwise Complement operator, works on one operand only and it returns the reverse of the output. That is, it negates the result.

1st Bit	~(1st Bit)
0	1
1	0

d. Bitwise XOR(^) : This operator results in false if the operands are of the same value.

1st Bit	2nd Bit	1st Bit ^ 2nd Bit
0	0	0
0	1	1
1	0	1
1	1	0

Some Solved Examples:

a. 6 & 3

First, we have to convert the given digits into their equivalent binary values.

6 = 0110 (In Binary)

3 = 0011 (In Binary)

Bit Operation of 6 & 3

```
  0110
& 0011
------
  0010
```

Which is 2 in decimal

Solution: 2

b. 10 | 7

10 = 1010 (In Binary)

7 = 0111 (In Binary)

Bit Operation of 10|7

```
   1010
|  0111
--------
00001111
```

Which is 15 in decimal

Solution: 15

c. ~5

5 = 0101 (In Binary)

~(5) = 1010 which is -5 in decimal

Solution: –5

d. 5 ^ 3

5 = 0101 (In Binary)

3 = 0011 (In Binary)

Bit Operation of 5 ^ 3

```
  0101
^ 0011
------
  0110
```

Which is 6 in decimal

Solution: 6

4. While naming a variable, it should follow certain protocols. Some of them are as follows:
   - It must start with a letter, an underscore (_) or a dollar ($) sign.
   - Apart from the first character, it can have any combination of characters including alphabets, digits and underscore only.
   - A variable name is case-sensitive, i.e., we can declare two different variables with different cases, e.g., int Sum, sum;
   - It can be of any length.
   - It cannot contain white space.
   - Reserved words cannot be used as variables in Java.
5. Ternary operator: A ternary operator works on three operands.

   condition ? expression1 : expression2

   Nested ternary is used to compare three numbers and find largest.
6. The assignment operator is used to assign a value or an expression to a variable. Java has one assignment operator which is the "=" operator. Whereas Java allows a way to represent some binary operators in a shorter way using shorthand assignment operators. They assign an expression to a variable. They are also known as compound assignment operators.

   One can say that Assignment (=) assigns value directly on the other side Shorthand (+=, -=, *=, /=) performs operation and assignment together

   Example:

   a = a + 5 → a += 5
7. Output of code is 12.

   Explanation:
   - Initial value: a = 5
   - Expression: b = a++ + ++a

   Step by step:

   1. a++ (post-increment)
      - oUses current value of a → 5
      - oThen a becomes 6
   2. ++a (pre-increment)
      - o First increases a from 6 to 7
      - o Then uses 7

   Now add the values:
   - b = 5 + 7 = 12

   So:
   - Final a = 7
   - Output printed: 12
8. Same as Ans. 6.

D. 1. The ternary (conditional) operator ?: is a compact alternative to an if-else statement. It evaluates a condition and returns one value if the condition is true, and another value if it is false.

Program using ternary operator (Java)

```
class DiscountCheck {
   public static void main(String[] args) {
      int age = 20;

      String result = (age > 18) ? "Eligible" : "Not Eligible";

      System.out.println(result);
   }
}
```

Advantages

- Short and concise code
- Improves readability for simple conditions
- Useful for quick assignments based on conditions
- Reduces number of lines in small logic checks

Limitations

- Becomes hard to read with complex conditions
- Not suitable for multiple conditions or nested decisions
- Can reduce clarity if overused
- Debugging is less straightforward compared to if-else

2. Given expression: int result = 10 + 20 * 5 / 2 - 3 % 4;

Step 1: Apply operator precedence

In Java, the precedence order here is:

1. *, /, % (same level, evaluated left to right)
2. +, - (same level, evaluated left to right)

Step 2: Evaluate step by step

Expression:

10 + 20 * 5 / 2 - 3 % 4

First handle *, /, % left to right:

- 20 * 5 = 100
- 100 / 2 = 50
- 3 % 4 = 3

Now expression becomes:

10 + 50 - 3

Next handle + and -:

- 10 + 50 = 60
- 60 - 3 = 57

Final answer:

result = 57

Changing the order using parentheses

If we want to change how the expression is evaluated, we use parentheses () because they have the highest precedence.

Example 1 (change grouping):

int result = (10 + 20) * 5 / (2 - 3 % 4);

Now evaluation changes:

- (10 + 20) = 30
- 3 % 4 = 3
- (2 - 3) = -1
- So expression becomes:

30 * 5 / -1 = 150 / -1 = -150

New result: -150

Why operator precedence matters

- It ensures expressions are evaluated in a predictable way
- Prevents logical errors in calculations
- Makes code consistent across different systems and compilers
- Helps write correct mathematical and algorithmic logic

E. 1. c. 2. b. 3. b. 4. d.

F. 1.

```
import java.util.Scanner;

public class DiscountCalculator {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the original price of the item: ");
        double price = sc.nextDouble();

        double discount = 0;
```

```
        double finalPrice = 0;

        if (price < 500) {
            discount = price * 0.05;
        }
        else if (price <= 1000) {
            discount = price * 0.10;
        }
        else {
            discount = price * 0.15;
        }

        finalPrice = price - discount;

        System.out.println("Original Price: " + price);
        System.out.println("Discount: " + discount);
        System.out.println("Final Price: " + finalPrice);
    }
}
```

2.
```
import java.util.Scanner;

public class CompareNumbers {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter first number: ");
        int a = sc.nextInt();

        System.out.print("Enter second number: ");
        int b = sc.nextInt();

        System.out.println("a == b : " + (a == b));
        System.out.println("a != b : " + (a != b));
        System.out.println("a > b : " + (a > b));
        System.out.println("a < b : " + (a < b));
```

```
            System.out.println("a >= b : " + (a >= b));
            System.out.println("a <= b : " + (a <= b));
        }
    }
```

3.
```
import java.util.Scanner;

public class LargestNumber {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter first number: ");
        int a = sc.nextInt();

        System.out.print("Enter second number: ");
        int b = sc.nextInt();

        System.out.print("Enter third number: ");
        int c = sc.nextInt();

        int largest = (a > b)
                      ? ((a > c) ? a : c)
                      : ((b > c) ? b : c);

        System.out.println("The largest number is: " + largest);
    }
}
```

G. Output of programs

1. a = 6
   b = 11
   Result = 22
2. x: 20
   y: 4
3. Result: true

8. Statements and Scope

MIND DRILL

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

B. 1. break 2. sequential 3. nested if 4. switch-case 5. branching
6. continue 7. Math.sqrt() 8. Math.ceil() 9. Double type
10. -21.0

C. 1. The scope of a variable defines the part of the program where it can be accessed. It determines the visibility and accessibility of a variable in Java.

2. Declaration statements are used to define the data type of variables or constants and to allocate memory for them.

3. Control statements are used to control the flow of execution of a program. They decide the direction in which statements are executed, such as sequential, conditional, iterative and jump flow.

4. In an if statement, the block executes only when the condition is true. In an if-else statement, one block executes if the condition is true and the other block executes if it is false.

5. A switch-case statement selects one block of code to execute from multiple cases based on the value of a variable or expression.

6. The break statement is used to terminate a case in a switch statement and exit the control structure, preventing fall-through to other cases.

7. Multiple conditions in switch-case are handled using different case labels for the same or different outputs, and the default case handles unmatched values.

8. A for loop is an entry-controlled loop used when the number of iterations is known in advance. It repeatedly executes a block of code for a fixed number of times.

9. A nested if statement is an if statement inside another if statement. It is useful when one condition depends on another.

   Example:

```
if(a > b)
{
if(a > c)
{
max = a;
}
else
```

```
{
max = c;
}
}
```

10. A switch-case statement is a multi-way selection structure that executes different blocks of code based on the value of an expression.

    Example:

```
switch(choice)
{
case 1: System.out.println("Good Morning"); break;
case 2: System.out.println("Good Afternoon"); break;
default: System.out.println("Good Night");
}
```

11. Fall Through occurs when break is not used in a switch-case, causing execution to continue into the next cases.

    Example:

```
switch(ch)
{
case 1: System.out.println(5+6);
case 2: System.out.println(5-4);
break;
case 3: System.out.println(5*4);
default: System.out.println(5.0/6.0);
}
```

12. Iterative statements are loops used to repeat execution of a block of code.

    •for loop: repeats a fixed number of times

    •while loop: repeats while condition is true

    •do-while loop: executes at least once, then repeats if condition is true

13. Entry-controlled loop checks condition before execution (for, while). Exit-controlled loop checks condition after execution (do-while).

    Example:

```
for(i=1; i<=5; i++)
{
System.out.println(i);
}
do
```

```
{
System.out.println(i);
i++;
}while(i<=5);
```

14.
```
import java.util.Scanner;
class PerfectNumber
{
public static void main(String args[])
{
Scanner sc = new Scanner(System.in);
int n = sc.nextInt();
int s = 0;
for(int i=1;i<=n/2;i++)
{
if(n%i==0)
s = s + i;
}
if(s==n)
System.out.println("Perfect number”);
else
System.out.println("Not a perfect number”);
}
}
```

15.
```
import java.util.Scanner;
class SumProduct
{
public static void main(String args[])
{
Scanner sc = new Scanner(System.in);
int n = sc.nextInt();
int sum=0, product=1, r;
while(n>0)
{
r = n%10;
sum = sum + r;
```

```
product = product * r;
n = n/10;
}
if(sum*product == n)
System.out.println("Sum product number”);
}
}
```

16. The continue statement is used to skip the current iteration of a loop and move to the next iteration.

    Program:

```
class EvenAverage
{
public static void main(String args[])
{
int sum=0, count=0;
for(int i=1;i<=20;i++)
{
if(i%2!=0)
continue;
sum = sum + i;
count++;
if(count==10)
break;
}
System.out.println("Average = " + (sum/10));
}
}
```

D. 1. We can handle borderline sentiment scores using a combination of if-else-if ladder and logical operators. First, define clear threshold ranges for Positive, Negative and Neutral. For overlapping values, we give priority rules. For example, if a score lies on a boundary, we check additional conditions like confidence level or exact score range using nested if statements. This ensures only one correct category is selected even when values overlap.

Example logic:

- if score > 0.6 → Positive
- else if score between -0.4 and 0.6 → Neutral
- else → Negative

If overlap occurs, nested if can refine decision (e.g., check confidence score or exact boundary value).

2. The control flow can be designed using if-else-if ladder for pricing + nested if for loyalty discount.

   First, decide price per user based on number of users:

   - if users ≤ 500 → 100
   - else if users ≤ 1500 → 400
   - else if users ≤ 5000 → 800
   - else → 1000

   Then calculate base cost:

   total = users × price

   Next apply loyalty discount using another if structure:

   - if months > 24 → 10% discount
   - else if months > 12 → 5% discount
   - else → no discount

   Final price:

   final = total - (total × discount / 100)

   This structure ensures:

   - First decision: pricing tier (using if-else-if ladder)
   - Second decision: discount rule (nested if)
   - Clear and sequential flow of control as per Java control statements from the chapter.

E. 1. a. The program didn't handle boundary conditions properly.

2. b. Modify the else if condition to use timeRequired >= 1 && timeRequired < 3.
3. a. if-else-if statement
4. b. Use a switch-case statement to handle different categories dynamically.

F. 1.

```
import java.util.Scanner;

class TaxiFare
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);

        int km;
        double bill = 0;
```

```
        System.out.print("Enter distance travelled: ");
        km = sc.nextInt();

        if(km <= 5)
        {
            bill = 75;
        }
        else if(km <= 15)
        {
            bill = 75 + (km - 5) * 15;
        }
        else if(km <= 25)
        {
            bill = 75 + (10 * 15) + (km - 15) * 10;
        }
        else
        {
            bill = 75 + (10 * 15) + (10 * 10) + (km - 25) * 5;
        }

        System.out.println("Total fare = " + bill);
    }
}
```

2. (i)
```
class Pattern1
{
    public static void main(String args[])
    {
        int i, j;

        for(i = 1; i <= 5; i++)
        {
            for(j = 1; j <= i; j++)
            {
                switch(j % 2)
                {
```

```
                    case 0: System.out.print("0 ");
                            break;
                    case 1: System.out.print("1 ");
                }
            }
            System.out.println();
        }
    }
}
```

(ii)

```
class Pattern2
{
    public static void main(String args[])
    {
        int i, j;

        for(i = 1; i <= 5; i++)
        {
            for(j = 1; j <= 5; j++)
            {
                switch(j >= i ? i : j)
                {
                    case 1:
                    case 2:
                    case 3:
                    case 4:
                    case 5:
                        if(j < i)
                            System.out.print(i + " ");
                        else
                            System.out.print(j + " ");
                        break;
                }
            }
            System.out.println();
        }
```

```
        }
    }
```

3.
```
import java.util.Scanner;

class PrimeFibonacci
{
    static boolean isPrime(int n)
    {
        if(n < 2) return false;

        for(int i = 2; i <= n/2; i++)
        {
            if(n % i == 0)
                return false;
        }
        return true;
    }

    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);

        int n;
        System.out.print("Enter limit: ");
        n = sc.nextInt();

        int a = 0, b = 1, c;

        while(a <= n)
        {
            if(isPrime(a))
            {
                System.out.print(a + " ");
            }
```

```
                c = a + b;
                a = b;
                b = c;
            }
        }
    }
```

1. 1
   2
   4
   8
2. Two
3. 10

H. 1. print all the even digits and odd digits

```
import java.util.Scanner;

class EvenOddDigits
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);

        int n, r;

        System.out.print("Enter a number: ");
        n = sc.nextInt();

        System.out.print("Even digits: ");
        int temp = n;

        while(temp > 0)
        {
            r = temp % 10;
            if(r % 2 == 0)
                System.out.print(r + " ");
            temp = temp / 10;
```

```
            }

            temp = n;

            System.out.print("\nOdd digits: ");
            while(temp > 0)
            {
                r = temp % 10;
                if(r % 2 != 0)
                    System.out.print(r + " ");
                temp = temp / 10;
            }
        }
    }
```

2. check whether it is divisible by 10 or not

```
import java.util.Scanner;

class DivisibleBy10
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        int n;
        System.out.print("Enter a number: ");
        n = sc.nextInt();
        if(n % 10 == 0)
            System.out.println("Divisible by 10”);
        else
            System.out.println("Not divisible by 10”);
    }
}
```

3. (a) 1 + 12 + 123 + ... + nth term

```
import java.util.Scanner;

class SeriesA
```

```
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);

        int n, i, j, term;

        System.out.print("Enter n: ");
        n = sc.nextInt();

        for(i = 1; i <= n; i++)
        {
            term = 0;
            for(j = 1; j <= i; j++)
            {
                term = term * 10 + j;
            }
            System.out.print(term + " ");
        }
    }
}
```

(b) S = a/1 + a/2 + c/3 + ... + a/n

```
import java.util.Scanner;

class SeriesB
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);

        int n, i, a = 1, c = 3;
        double sum = 0;

        System.out.print("Enter n: ");
        n = sc.nextInt();
```

```
            for(i = 1; i <= n; i++)
            {
                if(i % 2 == 1)
                    sum = sum + a / (double)i;
                else
                    sum = sum + c / (double)i;
            }

            System.out.println("Sum = " + sum);
        }
    }
```

(c) 1, 4, 9, 16, ... nth term

```
    import java.util.Scanner;

    class SeriesC
    {
        public static void main(String args[])
        {
            Scanner sc = new Scanner(System.in);

            int n, i;

            System.out.print("Enter n: ");
            n = sc.nextInt();

            for(i = 1; i <= n; i++)
            {
                System.out.print(i*i + " ");
            }
        }
    }
```

(d) 2 + 5 + 10 + 17 + ...

```
    import java.util.Scanner;
```

```
class SeriesD
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);

        int n, i, term = 2;

        System.out.print("Enter n: ");
        n = sc.nextInt();

        for(i = 1; i <= n; i++)
        {
            System.out.print(term + " ");
            term = term + i + 2;
        }
    }
}
```

(e) 1 * 2 * 3 * ... nth term

```
import java.util.Scanner;

class SeriesE
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);

        int n, i;
        long fact = 1;

        System.out.print("Enter n: ");
        n = sc.nextInt();

        for(i = 1; i <= n; i++)
        {
```

```
                fact = fact * i;
                System.out.print(fact + " ");
            }
        }
    }
```

4. Even sum & product of numbers ending with 4

```
import java.util.Scanner;

class EvenSumProduct
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);

        int n, num;
        int sumEven = 0;
        long productEnd4 = 1;
        boolean found = false;

        System.out.println("Enter 10 numbers:”);

        for(int i = 1; i <= 10; i++)
        {
            num = sc.nextInt();

            if(num % 2 == 0)
                sumEven = sumEven + num;

            if(num % 10 == 4)
            {
                productEnd4 = productEnd4 * num;
                found = true;
            }
        }
```

```
        System.out.println("Sum of even numbers = " + sumEven);

        if(found)
            System.out.println("Product of numbers ending with 4 =
" + productEnd4);
        else
            System.out.println("No number ends with 4”);
    }
}
```

5. Shopping mall discount program

```
import java.util.Scanner;

class DiscountCalc
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);

        String name;
        double amount, discount = 0, finalAmount;

        System.out.print("Enter customer name: ");
        name = sc.nextLine();

        System.out.print("Enter purchase amount: ");
        amount = sc.nextDouble();

        if(amount <= 2000)
            discount = 2.5;
        else if(amount <= 4000)
            discount = 4;
        else if(amount <= 7000)
            discount = 7;
        else
            discount = 10;
```

```
            finalAmount = amount - (amount * discount / 100);

            System.out.println("Customer Name: " + name);
            System.out.println("Discounted Amount: " + finalAmount);
        }
    }
```

6. Telephone bill program with surcharge

```
import java.util.Scanner;

class TelephoneBill
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);

        String name;
        int consumerNo, calls;
        double bill = 0, surcharge, totalBill;

        System.out.print("Enter consumer name: ");
        name = sc.nextLine();

        System.out.print("Enter consumer number: ");
        consumerNo = sc.nextInt();

        System.out.print("Enter number of calls: ");
        calls = sc.nextInt();

        if(calls > 250)
            bill = calls * 5;
        else if(calls >= 150)
            bill = calls * 4;
        else if(calls >= 75)
            bill = calls * 3;
```

```
            else
                bill = calls * 2;

            surcharge = bill * 2.5 / 100;
            totalBill = bill + surcharge;

            System.out.println("\nName of consumer: " + name);
            System.out.println("Consumer Number: " + consumerNo);
            System.out.println("Number of calls: " + calls);
            System.out.println("Bill amount before surcharge added: "
+ bill);
          System.out.println("Bill amount with surcharge: " + totalBill);
        }
}
```

7. Electronics Shop Discount

```
import java.util.Scanner;

class ElectronicsDiscount
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);

        String name, address, type;
        double amount, discountRate = 0, discount, netAmount;

        System.out.print("Enter customer name: ");
        name = sc.nextLine();

        System.out.print("Enter address: ");
        address = sc.nextLine();

        System.out.print("Enter purchase amount: ");
        amount = sc.nextDouble();
```

```
        System.out.print("Enter type of purchase (L for Laptop / D
for Desktop): ");
        type = sc.next();

        if(amount >= 0 && amount <= 25000)
        {
            if(type.equalsIgnoreCase("L”))
                discountRate = 0.0;
            else
                discountRate = 5.0;
        }
        else if(amount <= 57000)
        {
            if(type.equalsIgnoreCase("L”))
                discountRate = 5.0;
            else
                discountRate = 7.6;
        }
        else if(amount <= 100000)
        {
            if(type.equalsIgnoreCase("L”))
                discountRate = 7.5;
            else
                discountRate = 10.0;
        }
        else
        {
            if(type.equalsIgnoreCase("L”))
                discountRate = 10.0;
            else
                discountRate = 15.0;
        }

        discount = (discountRate / 100) * amount;
        netAmount = amount - discount;
```

```
            System.out.println("\nCustomer Name: " + name);
            System.out.println("Address: " + address);
            System.out.println("Purchase Amount: " + amount);
            System.out.println("Discount: " + discount);
            System.out.println("Net Amount to be Paid: " + netAmount);
        }
    }
```

8. Neon number / Palindrome number

```
import java.util.Scanner;

class NeonPalindrome
{
    static boolean isNeon(int n)
    {
        int sq = n * n;
        int sum = 0;

        while(sq > 0)
        {
            sum = sum + sq % 10;
            sq = sq / 10;
        }

        return sum == n;
    }

    static boolean isPalindrome(int n)
    {
        int rev = 0, temp = n;

        while(temp > 0)
        {
            rev = rev * 10 + temp % 10;
            temp = temp / 10;
        }
```

```
        return rev == n;
    }

    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);

        int choice, n;

        System.out.println("1. Check Neon Number”);
        System.out.println("2. Check Palindrome Number”);
        System.out.print("Enter choice: ");
        choice = sc.nextInt();

        System.out.print("Enter number: ");
        n = sc.nextInt();

        switch(choice)
        {
            case 1:
                if(isNeon(n))
                    System.out.println("Neon Number”);
                else
                    System.out.println("Not a Neon Number”);
                break;

            case 2:
                if(isPalindrome(n))
                    System.out.println("Palindrome Number”);
                else
                    System.out.println("Not a Palindrome Number”);
                break;

            default:
```

```
                System.out.println("Invalid Choice”);
            }
        }
    }
```

9. Automorphic number & GCD

```
import java.util.Scanner;

class AutomorphicGCD
{
    static boolean isAutomorphic(int n)
    {
        int sq = n * n;

        while(n > 0)
        {
            if(n % 10 != sq % 10)
                return false;

            n = n / 10;
            sq = sq / 10;
        }

        return true;
    }

    static int gcd(int a, int b)
    {
        while(b != 0)
        {
            int r = a % b;
            a = b;
            b = r;
        }
        return a;
    }
```

```
public static void main(String args[])
{
    Scanner sc = new Scanner(System.in);

    int choice;

    System.out.println("1. Check Automorphic Number”);
    System.out.println("2. Find GCD of two numbers”);
    System.out.print("Enter choice: ");
    choice = sc.nextInt();

    switch(choice)
    {
        case 1:
            System.out.print("Enter number: ");
            int n = sc.nextInt();

            if(isAutomorphic(n))
                System.out.println("Automorphic Number”);
            else
               System.out.println("Not an Automorphic Number”);
            break;

        case 2:
            System.out.print("Enter first number: ");
            int a = sc.nextInt();

            System.out.print("Enter second number: ");
            int b = sc.nextInt();

            System.out.println("GCD = " + gcd(a, b));
            break;

        default:
```

```
                System.out.println("Invalid Choice”);
            }
        }
    }
```

10. Pattern printing using switch case

```
import java.util.Scanner;

class PatternSwitch
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);

        int choice, i, j;

        System.out.println("1. Pattern A”);
        System.out.println("2. Pattern B”);
        System.out.print("Enter choice: ");
        choice = sc.nextInt();

        switch(choice)
        {
            case 1:
                for(i = 5; i >= 1; i--)
                {
                    for(j = 1; j <= i; j++)
                    {
                        System.out.print("*”);
                    }
                    System.out.println();
                }
                break;

            case 2:
                for(i = 1; i <= 5; i++)
```

```
                {
                    for(j = 1; j <= i; j++)
                    {
                        System.out.print("*”);
                    }
                    System.out.println();
                }
                break;

            default:
                System.out.println("Invalid Choice”);
        }
    }
}
```

11. Kaprekar & Fascinating number

```
import java.util.Scanner;

class KaprekarFascinating
{
    static boolean isKaprekar(int n)
    {
        if(n == 1) return true;

        int sq = n * n;
        String s = Integer.toString(sq);

        for(int i = 1; i < s.length(); i++)
        {
            String left = s.substring(0, i);
            String right = s.substring(i);

            int l = Integer.parseInt(left);
            int r = Integer.parseInt(right);

            if(r != 0 && l + r == n)
```

```
            return true;
        }
        return false;
    }

    static boolean isFascinating(int n)
    {
        String s = "" + n + (n * 2) + (n * 3);

        if(s.length() != 9)
            return false;

        for(char c = '1'; c <= '9'; c++)
        {
            int count = 0;

            for(int i = 0; i < s.length(); i++)
            {
                if(s.charAt(i) == c)
                    count++;
            }

            if(count != 1)
                return false;
        }
        return true;
    }

    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);

        int choice, n;

        System.out.println("1. Kaprekar Number");
```

```
            System.out.println("2. Fascinating Number");
            System.out.print("Enter choice: ");
            choice = sc.nextInt();

            System.out.print("Enter number: ");
            n = sc.nextInt();

            switch(choice)
            {
                case 1:
                    if(isKaprekar(n))
                        System.out.println("Kaprekar Number");
                    else
                        System.out.println("Not a Kaprekar Number");
                    break;

                case 2:
                    if(isFascinating(n))
                        System.out.println("Fascinating Number");
                    else
                       System.out.println("Not a Fascinating Number");
                    break;

                default:
                    System.out.println("Invalid Choice");
            }
        }
    }
```

12. Railway fare calculation (menu-based logic using nested if-else)

```
import java.util.Scanner;

class RailwayFare
{
    public static void main(String args[])
    {
```

```
Scanner sc = new Scanner(System.in);

int age, distance;
int fare = 0;

System.out.print("Enter age: ");
age = sc.nextInt();

System.out.print("Enter distance (in km): ");
distance = sc.nextInt();

if(age < 10)
{
    if(distance < 10)
        fare = 5;
    else if(distance <= 50)
        fare = 20;
    else
        fare = 50;
}
else if(age >= 10 && age <= 60)
{
    if(distance < 10)
        fare = 10;
    else if(distance <= 50)
        fare = 40;
    else
        fare = 80;
}
else
{
    if(distance < 10)
        fare = 4;
    else if(distance <= 50)
        fare = 15;
```

```
            else
                fare = 35;
        }

        System.out.println("Railway Fare = " + fare);
    }
}
```

13. Volume of cuboid, cylinder and cone (using switch case)

```
import java.util.Scanner;

class VolumeCalculator
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);

        int choice;
        double l, b, h, r, volume;
        double pi = 3.14;

        System.out.println("1. Cuboid”);
        System.out.println("2. Cylinder”);
        System.out.println("3. Cone”);

        System.out.print("Enter your choice: ");
        choice = sc.nextInt();

        switch(choice)
        {
            case 1:
                System.out.print("Enter length, breadth and
height: ");
                l = sc.nextDouble();
                b = sc.nextDouble();
                h = sc.nextDouble();
```

```
                volume = l * b * h;
                System.out.println("Volume of Cuboid = " + volume);
                break;

            case 2:
                System.out.print("Enter radius and height: ");
                r = sc.nextDouble();
                h = sc.nextDouble();

                volume = pi * r * r * h;
              System.out.println("Volume of Cylinder = " + volume);
                break;

            case 3:
                System.out.print("Enter radius and height: ");
                r = sc.nextDouble();
                h = sc.nextDouble();

                volume = (1.0 / 3) * pi * r * r * h;
                System.out.println("Volume of Cone = " + volume);
                break;

            default:
                System.out.println("Invalid Choice”);
        }
    }
}
```

14. Income tax calculation (male/female with slab rates)

```
import java.util.Scanner;

class IncomeTax
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
```

```
int income;
char gender;
double tax = 0;

System.out.print("Enter annual income: ");
income = sc.nextInt();

System.out.print("Enter gender (M/F): ");
gender = sc.next().charAt(0);

if(income <= 100000)
{
    tax = 0;
}
else if(income <= 200000)
{
    if(gender == 'F' || gender == 'f')
        tax = income * 10.0 / 100;
    else
        tax = income * 12.0 / 100;
}
else if(income <= 300000)
{
    if(gender == 'F' || gender == 'f')
        tax = income * 20.0 / 100;
    else
        tax = income * 22.0 / 100;
}
else if(income <= 500000)
{
    if(gender == 'F' || gender == 'f')
        tax = income * 30.0 / 100;
    else
        tax = income * 33.0 / 100;
}
```

```
        else
        {
            tax = income * 33.0 / 100;
        }

        System.out.println("Income Tax = " + tax);
    }
}
```

15. Pattern using switch case

```
import java.util.Scanner;

class Pattern15
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);

        int choice;
        System.out.println("1. Pattern I”);
        System.out.println("2. Pattern II”);
        choice = sc.nextInt();

        switch(choice)
        {
            case 1:
                int i, j, k, num = 1;

                for(i = 1; i <= 5; i++)
                {
                    for(j = 1; j <= i; j++)
                    {
                        if(num <= 15)
                            System.out.print(num + " ");
                        num++;
                    }
```

```
                System.out.println();
            }
            break;

        case 2:
            String s = "COMPUTER";

            for(i = 0; i < s.length(); i++)
            {
                for(j = 0; j < s.length() - i; j++)
                {
                    System.out.print(s.charAt(j));
                }
                System.out.println();
            }
            break;

        default:
            System.out.println("Invalid Choice");
    }
  }
}
```

16. Pattern using switch case

```
import java.util.Scanner;

class Pattern16
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);

        int choice;
        choice = sc.nextInt();

        switch(choice)
```

```
        {
            case 1:
                System.out.println("I”);
                System.out.println("I S”);
                System.out.println("I S C”);
                break;

            case 2:
                String s = "COMPUTER”;

                for(int i = 0; i < s.length(); i++)
                {
                    for(int j = 0; j < s.length() - i; j++)
                    {
                        System.out.print(s.charAt(j));
                    }
                    System.out.println();
                }
                break;

            default:
                System.out.println("Invalid Choice”);
        }
    }
}
```

17. Series programs (menu driven switch case)

```
import java.util.Scanner;

class Series
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);

        int choice, n;
```

```
double sum = 0, prod = 1;

System.out.println("1. 1/2 + 1/4 + ...");
System.out.println("2. Product series a/2 * a/3 ...");
System.out.println("3. Squares sum series");
System.out.println("4. 1/n^2 series");

choice = sc.nextInt();
n = sc.nextInt();

switch(choice)
{
    case 1:
        for(int i = 2; i <= n; i += 2)
            sum += 1.0 / i;
        System.out.println(sum);
        break;

    case 2:
        int a = 2;
        for(int i = 2; i <= n; i++)
            prod *= (double)a / i;
        System.out.println(prod);
        break;

    case 3:
        int s = 0;
        for(int i = 1; i <= n; i += 2)
            s += i;
        System.out.println(s);
        break;

    case 4:
        for(int i = 1; i <= n; i++)
            sum += 1.0 / (i * i);
```

```
                System.out.println(sum);
                break;

            default:
                System.out.println("Invalid Choice");
        }
    }
}
```

18. Prime Fibonacci numbers upto n

```
import java.util.Scanner;

class PrimeFibo
{
    static boolean isPrime(int n)
    {
        if(n < 2) return false;

        for(int i = 2; i <= n/2; i++)
            if(n % i == 0)
                return false;

        return true;
    }

    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);

        int n = sc.nextInt();
        int a = 0, b = 1, c;

        while(a <= n)
        {
            if(isPrime(a))
                System.out.print(a + " ");
```

```
            c = a + b;
            a = b;
            b = c;
        }
    }
}
```

19. Switch case series

```
import java.util.Scanner;

class SeriesSwitch
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);

        int choice = sc.nextInt();

        switch(choice)
        {
            case 1:
                for(int i = 1, s = 1; i <= 10; i++)
                {
                    System.out.print(s + " ");
                    s += i + 1;
                }
                break;

            case 2:
                for(int i = 1; i <= 10; i++)
                    System.out.print(i*i + " ");
                break;

            case 3:
                for(int i = 1; i <= 10; i++)
```

```
                    System.out.print(i*i*i + " ");
                break;
            }
        }
    }
```

20. Prime perfect number (all factors are prime)

```
import java.util.Scanner;

class PrimePerfect
{
    static boolean isPrime(int n)
    {
        if(n < 2) return false;

        for(int i = 2; i <= n/2; i++)
            if(n % i == 0)
                return false;

        return true;
    }

    static boolean isPrimePerfect(int n)
    {
        for(int i = 1; i <= n; i++)
        {
            if(n % i == 0)
            {
                if(!isPrime(i))
                    return false;
            }
        }
        return true;
    }

    public static void main(String args[])
```

```
    {
        Scanner sc = new Scanner(System.in);

        int n = sc.nextInt();

        for(int i = 1; i <= n; i++)
        {
            if(isPrimePerfect(i))
                System.out.print(i + " ");
        }
    }
}
```

21. Grade and remark program

```
import java.util.Scanner;

class GradeSystem
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);

        int m1, m2, m3, total;
        double avg;

        m1 = sc.nextInt();
        m2 = sc.nextInt();
        m3 = sc.nextInt();

        total = m1 + m2 + m3;
        avg = total / 3.0;

        System.out.println("Total = " + total);

        if(avg >= 85)
            System.out.println("Grade A - Excellent”);
```

```
            else if(avg >= 75)
                System.out.println("Grade B - Distinction”);
            else if(avg >= 60)
                System.out.println("Grade C - Good”);
            else if(avg >= 40)
                System.out.println("Grade D - Fair”);
            else
                System.out.println("Grade E - Need Improvement”);
        }
    }
```

9. Methods and Constructors

MIND DRILL

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

B. 1. constructor 2. static 3. instance variable 4. data members
5. data members and member functions
6. new 7. static data member 8. impure methods

C. 1. The method header is the first line of a method definition which contains the access specifier, return type, method name and parameter list, whereas the method signature consists only of the method name along with its parameter list which uniquely identifies a method.

2. Methods reduce code repetition, make debugging easier, reduce program size and improve execution efficiency and memory usage.

3. The general syntax of declaring a method is <access specifier> <return type> <method name> (parameter list) { body of method }

4. The return type specifies the type of value a method returns or void if no value is returned, whereas the return statement is used to return a value from a method and transfer control back to the caller.

5. The return statement is the last statement in a method and terminates the method execution, and only one value can be returned at a time.

6. Method overloading is the process of having multiple methods with the same name but different parameter lists, and the compiler differentiates them based on the number and type of parameters.

7. Access specifiers are keywords that define the accessibility of methods and variables; public allows access from anywhere, private allows access only within the same class, and protected allows access within the same class and its subclasses.
8. Actual parameters are the values passed during a method call, whereas formal parameters are the variables defined in the method that receive those values.

D. 1. The first method is a pure method because it does not change the original value passed to it and only returns a result based on input. It has no side effects.

The second method is an impure method because it modifies the array passed to it. It produces side effects by changing the original data.

2. Constructors do not have a return type because their main purpose is to initialise an object, not to return any value. Even void is not used since constructors are automatically invoked when an object is created using the new operator. This ensures proper object initialisation at the time of creation without requiring a method call.

E. 1. c. 2. b. 3. b. 4. b. 5. c.

F. 1.

```
class Geometry
{
    void geometry(int n, char ch)
    {
        for(int i = 1; i <= n; i++)
        {
            for(int j = 1; j <= n; j++)
            {
                System.out.print(ch);
            }
            System.out.println();
        }
    }

    void geometry(int x, int y)
    {
        for(int i = 1; i <= y; i++)
        {
            for(int j = 1; j <= x; j++)
            {
                System.out.print('a');
            }
            System.out.println();
```

```
        }
    }

    void geometry()
    {
        int rows = 4;

        for(int i = 1; i <= rows; i++)
        {
            for(int j = 1; j <= (2 * i - 1); j++)
            {
                System.out.print("$”);
            }
            System.out.println();
        }
    }

    public static void main(String args[])
    {
        Geometry obj = new Geometry();

        obj.geometry(3, ‘x’);
        System.out.println();

        obj.geometry(3, 4);
        System.out.println();

        obj.geometry();
    }
}
```

2.
```
class Area
{
    double area(double a, double b, double c)
    {
        double s = (a + b + c) / 2;
```

```
        return Math.sqrt(s * (s - a) * (s - b) * (s - c));
    }

    double area(int a, int b, int height)
    {
        return 0.5 * height * (a + b);
    }

    double area(double d1, double d2)
    {
        return 0.5 * d1 * d2;
    }

    public static void main(String args[])
    {
        Area obj = new Area();

        System.out.println("Area of Scalene Triangle: " + obj.
area(3.0, 4.0, 5.0));
        System.out.println("Area of Trapezium: " + obj.area(4, 6,
5));
       System.out.println("Area of Rhombus: " + obj.area(4.0, 6.0));
    }
}
```

3.
```
class Series
{
    int series(int n)
    {
        int sum = 0;

        for(int i = 1; i <= n; i++)
        {
            sum = sum + (i * i);
        }
```

```
    return sum;
}

void series(int x, int n)
{
    int sum = 0;

    for(int i = 1; i <= n; i++)
    {
        sum = sum + (x - (2 * i - 1));
    }

    System.out.println("Sum of series: " + sum);
}

double series(double n)
{
    double sum = 0;

    for(int i = 1; i <= n; i++)
    {
        sum = sum + (i * i);
        System.out.print((i * i) + " ");
    }

    System.out.println();
    return sum;
}

public static void main(String args[])
{
    Series obj = new Series();

  System.out.println("Series sum (1^2 to n^2): " + obj.series(5));
```

```
        obj.series(10, 4);

        System.out.println("Sum returned: " + obj.series(5.0));
    }
}
```

G. 1. Sum: 40 and Average: 20.0
Sum: 55 and Average: 27.5

2. Error

3. Counter:0
Counter:1
Counter:2

4. Name: Ajay
Basic Salary: 40000.0
HRA: 4000.0
DA: 22000.0
PF: 3332.0
Gross Salary: 66000.0
Net Salary: 62668.0

H. 1.
```
class Guard
{
    String gn;
    int hr;
    double rate, wg;

    Guard(String gn, int hr)
    {
        this.gn = gn;
        this.hr = hr;
    }
    void calcwage()
    {
        if(hr <= 48)
            rate = 3;
        else if(hr <= 53)
            rate = 4;
```

```
            else if(hr <= 55)
                rate = 5;
            else
            {
                rate = 0;
                System.out.println("No overtime allowed”);
            }
            wg = hr * rate;
        }

        void display()
        {
            System.out.println("Name\tHours\tWage”);
            System.out.println(gn + "\t” + hr + "\t” + wg);
        }
        public static void main(String args[])
        {
            Guard g = new Guard("Amit”, 50);
            g.calcwage();
            g.display();
        }
    }
```

2.
```
    class Triplet
    {
        int n1, n2, n3;
        Triplet(int a, int b, int c)
        {
            n1 = a;
            n2 = b;
            n3 = c;
        }
        void check()
        {
            if(n1*n1 + n2*n2 == n3*n3)
```

```
            System.out.println("Pythagorean Triplet”);
        else
            System.out.println("Not a Triplet”);
    }
    public static void main(String args[])
    {
        Triplet t = new Triplet(3,4,5);
        t.check();
    }
}
```

3.
```
class Shop
{
    String item;
    int stock;

    Shop(String i, int s)
    {
        item = i;
        stock = s;
    }
    void buy(int n)
    {
        if(stock >= n)
        {
            stock -= n;
            System.out.println("Purchase successful. Remaining: "
+ stock);
        }
        else
            System.out.println("Not enough stock”);
    }
    public static void main(String args[])
    {
        Shop s = new Shop("Pen”, 10);
        s.buy(3);
```

```
        }
    }
```

4.
```
class Perfect
{
    int n;

    Perfect(int n)
    {
        this.n = n;
    }
    void displaySquares()
    {
        int count = 0, i = n + 1;
        while(count < 5)
        {
            double r = Math.sqrt(i);
            if(r == (int)r)
            {
                System.out.print(i + " ");
                count++;
            }
            i++;
        }
    }
    public static void main(String args[])
    {
        Perfect p = new Perfect(15);
        p.displaySquares();
    }
}
```

5.
```
class Happy
{
    int n;
    Happy(int n)
    {
```

```
        this.n = n;
    }

    int sumSq(int x)
    {
        int sum = 0;
        while(x > 0)
        {
            int d = x % 10;
            sum += d * d;
            x /= 10;
        }
        return sum;
    }
    void check()
    {
        int temp = n;
        while(temp != 1 && temp != 4)
            temp = sumSq(temp);

        if(temp == 1)
            System.out.println("Happy Number”);
        else
            System.out.println("Not Happy Number”);
    }
    public static void main(String args[])
    {
        Happy h = new Happy(19);
        h.check();
    }
}
```

6.
```
class Student
{
    String name;
    double m, p, c;
```

```
    Student(String name, double m, double p, double c)
    {
        this.name = name;
        this.m = m;
        this.p = p;
        this.c = c;
    }
    void check()
    {
        double avg = (m + p + c) / 3;

        if(avg >= 90)
            System.out.println(name + " Pure Science with CS”);
        else if(m >= 90 && p >= 90)
            System.out.println(name + " Bio Science”);
        else if(avg >= 70)
            System.out.println(name + " Commerce”);
        else
            System.out.println(name + " Humanities”);
    }

    public static void main(String args[])
    {
        Student s = new Student("Ravi”, 92, 88, 95);
        s.check();
    }
}
```

7.
```
class Loan
{
    int p, t;
    double rate, interest;

    void accept(int p, int t)
    {
        this.p = p;
```

```
        this.t = t;
    }
    void calc()
    {
        if(t == 1) rate = 4.5;
        else if(t == 2) rate = 5;
        else if(t == 3) rate = 7.5;
        else rate = 10;

        interest = p * rate / 100;
    }
    void display()
    {
        System.out.println("Interest: " + interest);
    }
    public static void main(String args[])
    {
        Loan l = new Loan();
        l.accept(10000, 3);
        l.calc();
        l.display();
    }
}
```

8.
```
class PalPrime
{
    boolean isPrime(int n)
    {
        for(int i = 2; i < n; i++)
            if(n % i == 0) return false;
        return true;
    }

    boolean isPal(int n)
    {
        int r = 0, t = n;
```

```
        while(t > 0)
        {
            r = r * 10 + t % 10;
            t /= 10;
        }
        return r == n;
    }
    void display()
    {
        for(int i = 10; i <= 100; i++)
        {
            if(isPrime(i) && isPal(i))
                System.out.print(i + " ");
        }
    }
    public static void main(String args[])
    {
        PalPrime p = new PalPrime();
        p.display();
    }
}
```

9.
```
class BookFair
{
    String name;
    double price;

    BookFair(String n, double p)
    {
        name = n;
        price = p;
    }
    void calculate()
    {
        double d;
```

```
            if(price <= 1000)
                d = price * 2 / 100;
            else if(price <= 3000)
                d = price * 10 / 100;
            else
                d = price * 15 / 100;

            price -= d;
        }
        void display()
        {
            System.out.println(name + " Final Price: " + price);
        }
        public static void main(String args[])
        {
            BookFair b = new BookFair("Math Book”, 1500);
            b.calculate();
            b.display();
        }
    }
```

10.
```
    class Library
    {
        int acc;
        String title, author;
        void input(int a, String t, String au)
        {
            acc = a;
            title = t;
            author = au;
        }
        void display(int days)
        {
            int fine = days * 2;
            System.out.println(acc + " " + title + " " + author + "
    Fine: " + fine);
```

```
    }
    public static void main(String args[])
    {
        Library l = new Library();
        l.input(101, "Java", "James");
        l.display(5);
    }
}
```

11.
```
class Discount
{
    String name;
    int cost;
    double dc, amt;
    void accept(String n, int c)
    {
        name = n;
        cost = c;
    }
    void calc()
    {
        if(cost <= 5000)
            dc = 0;
        else if(cost <= 10000)
            dc = cost * 10 / 100;
        else if(cost <= 15000)
            dc = cost * 15 / 100;
        else
            dc = cost * 20 / 100;
        amt = cost - dc;
    }
    void display()
    {
        System.out.println(name + " Discount: " + dc + " Pay: " +
amt);
    }
```

```
    public static void main(String args[])
    {
        Discount d = new Discount();
        d.accept("Rahul”, 12000);
        d.calc();
        d.display();
    }
}
```

12.
```
class SeriesSum
{
    int fact(int n)
    {
        int f = 1;
        for(int i = 1; i <= n; i++)
            f *= i;
        return f;
    }
    double power(int x, int y)
    {
        return Math.pow(x, y);
    }
    double calc(int x, int n)
    {
        double s = 0;
        for(int i = 1; i <= n; i++)
            s += power(x, i) / fact(i);
        return s;
    }
    public static void main(String args[])
    {
        SeriesSum s = new SeriesSum();
        System.out.println(s.calc(2, 5));
    }
}
```

13.
```
class Compare
```

```
{
    void compare(int a, int b)
    {
        System.out.println(Math.max(a, b));
    }
    void compare(char a, char b)
    {
        System.out.println((a > b) ? a : b);
    }
    void compare(String a, String b)
    {
        System.out.println((a.length() > b.length()) ? a : b);
    }
    public static void main(String args[])
    {
        Compare c = new Compare();
        c.compare(10, 20);
        c.compare('A', 'Z');
        c.compare("Java", "Python");
    }
}
```

10. Strings

MIND DRILL

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

B. 1. true 2. 0 3. countTokens() 4. uppercase
5. smaller, greater

C. 1. The String class is used to store and manipulate a sequence of characters enclosed in double quotes. It provides various methods to perform operations like finding length, extracting substrings, comparing strings, changing case, and joining strings.

2. The StringBuffer class is used to create and manage mutable strings, meaning the content of the string can be changed after creation without creating a new object.

3. The StringTokenizer class is used to break a string into smaller parts called tokens based on delimiters such as spaces or special characters.
4. The Character class is a built-in Java class used to wrap a single char value and provides methods to manipulate and test characters.

   Some methods include:

   - isLetter(): checks if a character is an alphabet
   - isDigit(): checks if a character is a number
   - isUpperCase(): checks if a character is uppercase
   - isLowerCase(): checks if a character is lowercase
   - toUpperCase(): converts a character to uppercase
   - toLowerCase(): converts a character to lowercase
   - isLetterOrDigit(): checks if it is a letter or digit
   - isWhitespace(): checks for space characters
5. The length() function is used to find the total number of characters in a string. It returns an integer value and counts all letters, digits, spaces, and symbols in the string.
6. The indexOf(char ch, int index) function returns the position of a character starting the search from a given index, whereas lastIndexOf(char ch) returns the position of the last occurrence of a character in the string.
7. The replace(char charToReplace, char charToInsert) function is used when we need to replace all occurrences of a particular character in a string with another character, such as correcting spelling mistakes or modifying text data.
8. The append() method is used in the StringBuffer class to add one string at the end of another string. It modifies the existing object by extending it with the new string instead of creating a new object.

D. 1.
```
class StringDemo1
{
    public static void main(String[] args)
    {
        String str = "Java Programming Language";

        System.out.println("Length : " + str.length());
        System.out.println("Index of g : " + str.indexOf('g'));
        System.out.println(str.toUpperCase());
    }
}
```

2.
```
class StringDemo2
{
```

```
    public static void main(String[] args)
    {
        String str = "Welcome To Computer World";

        System.out.println(str.toLowerCase());
        System.out.println("Index of o from index 10 : " + str.
indexOf('o', 10));
        System.out.println("Length : " + str.length());
    }
}
```

3.
```
class StringBufferDemo
{
    public static void main(String[] args)
    {
        StringBuffer sb = new StringBuffer("Technology and
Innovation Drive Growth");

        // Delete "and " (starting index 11 to 15)
        sb.delete(11, 15);

        // Convert to String and uppercase
        String result = sb.toString().toUpperCase();

        System.out.println(result);
    }
```

E. 1. b. It removes a range of characters from the string

2. b. Modifies the character at a specific index

3. a. "Java ming"

F. 1.
```
class StringDemo1
{
    public static void main(String[] args)
    {
        String str = "Java Programming Language";

        System.out.println("Length : " + str.length());
```

```
            System.out.println("Index of g : " + str.indexOf('g'));
            System.out.println(str.toUpperCase());
        }
    }
```

2.
```
class StringDemo2
{
    public static void main(String[] args)
    {
        String str = "Welcome To Computer World";

        System.out.println(str.toLowerCase());
        System.out.println("Index of o from index 10 : " + str.
indexOf('o', 10));
        System.out.println("Length : " + str.length());
    }
}
```

3.
```
class StringBufferDemo
{
    public static void main(String[] args)
    {
        StringBuffer sb = new StringBuffer("Technology and
Innovation Drive Growth");

        // Delete "and " (starting index 11 to 15)
        sb.delete(11, 15);

        // Convert to String and uppercase
        String result = sb.toString().toUpperCase();

        System.out.println(result);
    }
}
```

G. 1. Length : 9
Index of 'a' : 5
EDUCATION

2. character123
   Index of 'r' from index 4 : 6
   Length : 13
3. Modern

11. Arrays

MIND DRILL

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

B. 1. 0 2. Linear 3. Merging 4. comma (,) 5. Scanner

C. 1. You can access the first element of an array using index 0, for example arr[0].

2. A 2D array is an array of arrays, where data is stored in rows and columns.

3. Sorting is the process of arranging array elements in ascending or descending order.

4. Bubble sort is a sorting technique in which adjacent elements are compared and swapped repeatedly until the array is sorted.

5. In static array declaration, the size and values are fixed at compile time. Example:

   int ar[] = {1, 2, 3};

   In dynamic array declaration, the size is given at runtime using new. Example:

   int ar[] = new int[5];

6. Insertion in an array means adding a new element at a specific position by shifting existing elements to the right.

   Example Java program:

```
import java.util.*;
class InsertArray
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        int ar[] = new int[6];
        int i, pos, val;

        System.out.println("Enter 5 elements:”);
```

```
        for(i = 0; i < 5; i++)
        {
            ar[i] = sc.nextInt();
        }

        System.out.println("Enter position and value to insert:”);
        pos = sc.nextInt();
        val = sc.nextInt();

        for(i = 4; i >= pos; i--)
        {
            ar[i+1] = ar[i];
        }

        ar[pos] = val;

        System.out.println("Array after insertion:”);
        for(i = 0; i < 6; i++)
        {
            System.out.print(ar[i] + " ");
        }
    }
}
```

7. Difference between bubble sort and selection sort:

   Bubble sort repeatedly swaps adjacent elements if they are in wrong order, while selection sort repeatedly selects the smallest (or largest) element and places it in correct position. Bubble sort involves more swaps, selection sort involves fewer swaps.

8. Merging two arrays in Java means combining elements of two arrays into a single array. The elements of the first array are copied first, followed by the elements of the second array.

9. Linear search is a method of searching where each element is checked one by one from the beginning to the end until the target is found.

   Java program for linear search:

```
import java.util.*;
class LinearSearch
{
    public static void main(String args[])
```

```
    {
        Scanner sc = new Scanner(System.in);
        int n, i, key, pos = -1;

        System.out.println("Enter size:”);
        n = sc.nextInt();

        int ar[] = new int[n];

        System.out.println("Enter elements:”);
        for(i = 0; i < n; i++)
        {
            ar[i] = sc.nextInt();
        }

        System.out.println("Enter element to search:”);
        key = sc.nextInt();

        for(i = 0; i < n; i++)
        {
            if(ar[i] == key)
            {
                pos = i;
                break;
            }
        }

        if(pos != -1)
            System.out.println("Found at index " + pos);
        else
            System.out.println("Not found”);
    }
}
```

D. 1. To solve this, we can use a simple greedy approach:

Idea:

Track the minimum price seen so far and calculate the maximum profit at each step.

Steps:

1. Assume the first day price is the minimum price.
2. Traverse the array from left to right.
3. At each day:
   - Update the minimum price if current price is lower.
   - Calculate profit = current price − minimum price.
   - Update maximum profit if this profit is higher.
4. At the end, the maximum profit gives the best buy-sell pair.

2. Most efficient approach: Boyer–Moore Voting Algorithm

Steps:

1. Assume first element is the candidate.
2. Set count = 1.
3. Traverse the array:
   - If same element → increase count.
   - If different → decrease count.
   - If count becomes 0 → change candidate.
4. After finding candidate, do a second pass to confirm it appears more than n/2 times.

   Time Complexity:
   - O(n) (two linear passes)
   - Space Complexity: O(1)

E. 1. a. Arrays can only store data of one type

2. a. ArrayList
3. b. Iterate through the array and keeping track of the highest one
4. b. ArrayLists automatically adjust their size

F. 1.

```
class Employee
{
    public static void main(String args[])
    {
        String name[] = {"John", "Sara", "Mike"};
        int age[] = {25, 30, 28};

        System.out.println("ID Name Age");
```

```
            for(int i = 0; i < 3; i++)
            {
                System.out.println(i + " " + name[i] + " " + age[i]);
            }
        }
    }
```

2.
```
    class SumAverage
    {
        public static void main(String args[])
        {
            int ar[] = {10, 20, 30, 40, 50};
            int sum = 0;

            System.out.print("Array Elements: ");

            for(int i = 0; i < ar.length; i++)
            {
                System.out.print(ar[i] + " ");
                sum = sum + ar[i];
            }

            double avg = sum / 5.0;

            System.out.println("\nSum: " + sum);
            System.out.println("Average: " + avg);
        }
    }
```

3.
```
    class RowMax
    {
        public static void main(String args[])
        {
            int ar[][] = {
                {10, 20, 30},
                {15, 25, 5},
```

```
            {40, 50, 60}
        };

        for(int i = 0; i < 3; i++)
        {
            int max = ar[i][0];

            for(int j = 1; j < 3; j++)
            {
                if(ar[i][j] > max)
                {
                    max = ar[i][j];
                }
            }

            System.out.println("Row " + (i + 1) + ": Highest: " +
max);
        }
    }
}
```

4.
```
class MissingNumber
{
    public static void main(String args[])
    {
        int ar[] = {1, 2, 3, 5, 6};
        int n = 6;

        int total = n * (n + 1) / 2;
        int sum = 0;

        for(int i = 0; i < ar.length; i++)
        {
            sum = sum + ar[i];
        }
```

```
            int missing = total - sum;

            System.out.println("The missing number is " + missing);
        }
    }
```

G. 1. 5

2. Total marks: 355
   Average marks: 71.0

3. 30 found at position 3

H. 1.
```
class OddAndMultiple7
{
    public static void main(String args[])
    {
        int ar[] = {1, -3, 5, 7, 14, -21, 28, 9, 11, -7};
        int sumOdd = 0;
        long product7 = 1;
        int found = 0;
        for(int i = 0; i < ar.length; i++)
        {
            if(ar[i] % 2 != 0)
                sumOdd = sumOdd + ar[i];
            if(ar[i] % 7 == 0)
            {
                product7 = product7 * ar[i];
                found = 1;
            }
        }
        if(found == 0)
            product7 = 0;
        System.out.println("Sum of odd numbers: " + sumOdd);
       System.out.println("Product of multiples of 7: " + product7);
    }
}
```

2.
```
import java.util.*;
```

```
class BubbleSortNames
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        String ar[] = new String[20];
        System.out.println("Enter 20 names:”);
        for(int i = 0; i < 20; i++)
        {
            ar[i] = sc.next();
        }
        for(int i = 0; i < 19; i++)
        {
            for(int j = 0; j < 19 - i; j++)
            {
                if(ar[j].compareTo(ar[j+1]) > 0)
                {
                    String temp = ar[j];
                    ar[j] = ar[j+1];
                    ar[j+1] = temp;
                }
            }
        }

        System.out.println("Sorted names:”);

        for(int i = 0; i < 20; i++)
        {
            System.out.println(ar[i]);
        }
    }
}
```

3.
```
class InsertionSortDesc
{
    public static void main(String args[])
```

```
    {
        int ar[] = {12, 45, 3, 89, 22, 67, 90, 11, 5, 33,
                    14, 56, 78, 9, 41, 60, 72, 18, 27, 99,
                    31, 44, 8, 2, 50};
        for(int i = 1; i < ar.length; i++)
        {
            int key = ar[i];
            int j = i - 1;
            while(j >= 0 && ar[j] < key)
            {
                ar[j + 1] = ar[j];
                j--;
            }
            ar[j + 1] = key;
        }
        for(int i = 0; i < ar.length; i++)
        {
            System.out.print(ar[i] + " ");
        }
    }
}
```

4.
```
import java.util.*;

class PrimeArray
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        int ar[] = new int[10];
        System.out.println("Enter 10 numbers:”);
        for(int i = 0; i < 10; i++)
        {
            ar[i] = sc.nextInt();
        }
        System.out.println("Prime numbers:”);
```

```
        for(int i = 0; i < 10; i++)
        {
            int num = ar[i];
            int count = 0;
            for(int j = 1; j <= num; j++)
            {
                if(num % j == 0)
                    count++;
            }
            if(count == 2)
                System.out.println(num);
        }
    }
}
```

5.
```
import java.util.*;
class SwitchArray
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        int ar[] = new int[5];

        System.out.println("Enter 5 numbers:”);
        for(int i = 0; i < 5; i++)
            ar[i] = sc.nextInt();
        System.out.println("Enter choice (1 or 2):”);
        int ch = sc.nextInt();
        switch(ch)
        {
            case 1:
                int sum = 0;
                for(int i = 0; i < 5; i++)
                {
                    int num = ar[i];
                    int count = 0;
```

```
                    for(int j = 1; j <= num; j++)
                    {
                        if(num % j == 0)
                            count++;
                    }
                    if(count == 2)
                        sum = sum + num;
                }
                System.out.println("Sum of prime numbers: " + sum);
                break;

            case 2:
                System.out.println("Odd negative numbers:”);
                for(int i = 0; i < 5; i++)
                {
                    if(ar[i] < 0 && ar[i] % 2 != 0)
                        System.out.println(ar[i]);
                }
                break;
        }
    }
}
```

6.
```
class Factorial2D
{
    public static void main(String args[])
    {
        int ar[][] = {{2, 3, 4}, {5, 6, 7}};

        for(int i = 0; i < 2; i++)
        {
            for(int j = 0; j < 3; j++)
            {
                int num = ar[i][j];
                int fact = 1;
                for(int k = 1; k <= num; k++)
```

```
                    fact = fact * k;
                System.out.println("Factorial of " + num + " = " +
fact);
            }
        }
    }
}
```

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

class Items
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        String name[] = new String[5];
        double price[] = new double[5];
        double total = 0;
        for(int i = 0; i < 5; i++)
        {
            System.out.println("Enter item name:”);
            name[i] = sc.next();
            System.out.println("Enter price:”);
            price[i] = sc.nextDouble();
            total = total + price[i];
        }
        System.out.println("Item Name\tPrice”);

        for(int i = 0; i < 5; i++)
        {
            System.out.println(name[i] + "\t\t” + price[i]);
        }

        System.out.println("Total Amount: " + total);
    }
}
```

8.
```
import java.util.*;
class OddEvenArray
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        int ar[] = new int[10];
        int even[] = new int[10];
        int odd[] = new int[10];
        int e = 0, o = 0;
        System.out.println("Enter numbers:”);
        for(int i = 0; i < 10; i++)
        {
            ar[i] = sc.nextInt();

            if(ar[i] % 2 == 0)
                even[e++] = ar[i];
            else
                odd[o++] = ar[i];
        }
        System.out.println("Even numbers:”);
        for(int i = 0; i < e; i++)
            System.out.println(even[i]);
        System.out.println("Odd numbers:”);
        for(int i = 0; i < o; i++)
            System.out.println(odd[i]);
    }
}
```

9.
```
import java.util.*;

class Merit
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
```

```
        String name[] = new String[40];
        int marks[] = new int[40];
        System.out.println("Enter names and marks:”);

        for(int i = 0; i < 40; i++)
        {
            name[i] = sc.next();
            marks[i] = sc.nextInt();
        }

        for(int i = 0; i < 39; i++)
        {
            for(int j = 0; j < 39 - i; j++)
            {
                if(marks[j] < marks[j+1])
                {
                    int temp = marks[j];
                    marks[j] = marks[j+1];
                    marks[j+1] = temp;

                    String t = name[j];
                    name[j] = name[j+1];
                    name[j+1] = t;
                }
            }
        }
        System.out.println("Merit List:”);

        for(int i = 0; i < 40; i++)
        {
            System.out.println(name[i] + " " + marks[i]);
        }
    }
}
```

10.
```
class CommonElements
{
    public static void main(String args[])
    {
        int A[] = {1, 2, 3, 4, 5};
        int B[] = {3, 4, 5, 6, 7};
        System.out.println("Common elements:”);
        for(int i = 0; i < A.length; i++)
        {
            for(int j = 0; j < B.length; j++)
            {
                if(A[i] == B[j])
                {
                    System.out.println(A[i]);
                }
            }
        }
    }
}
```

11.
```
import java.util.*;
class CountryCity
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        String country[] = {"GERMANY”, "NEPAL”, "JAPAN”, "CANADA”,
"IRAQ”};
        String city[] = {"BERLIN”, "KATMANDU”, "TOKYO”,
"MONTREAL”, "BAGHDAD”};
        while(true)
        {
           System.out.println("Enter country name (XXX to stop):”);
            String input = sc.next();
            if(input.equals("XXX”))
                break;
```

```
                int pos = -1;
                for(int i = 0; i < country.length; i++)
                {
                    if(country[i].equalsIgnoreCase(input))
                    {
                        pos = i;
                        break;
                    }
                }
                if(pos != -1)
                    System.out.println("City: " + city[pos]);
                else
                    System.out.println("Error: Country not found”);
            }
        }
    }
```

12.
```
import java.util.*;

class StudentAverage
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        int roll[] = new int[50];
        int A[] = new int[50];
        int B[] = new int[50];
        int C[] = new int[50];
        for(int i = 0; i < 50; i++)
        {
            System.out.println("Enter Roll No and marks of A, B,
C:”);
            roll[i] = sc.nextInt();
            A[i] = sc.nextInt();
            B[i] = sc.nextInt();
            C[i] = sc.nextInt();
```

```
            }

            for(int i = 0; i < 50; i++)
            {
                double avg = (A[i] + B[i] + C[i]) / 3.0;
                if(avg > 80)
                {
                    System.out.println("Roll No: " + roll[i]);
                    System.out.println("Average: " + avg);
                }
            }
        }
    }
```

13.
```
    import java.util.*;

    class Politicians
    {
        public static void main(String args[])
        {
            Scanner sc = new Scanner(System.in);
            String name[] = new String[10];
            String party[] = new String[10];
            for(int i = 0; i < 10; i++)
            {
                System.out.println("Enter politician name and party:”);
                name[i] = sc.next();
                party[i] = sc.next();
            }
            while(true)
            {
                System.out.println("Enter party name (STOP to end):”);
                String p = sc.next();

                if(p.equalsIgnoreCase("STOP”))
                    break;
```

```
                System.out.println("Politicians:”);

                for(int i = 0; i < 10; i++)
                {
                    if(party[i].equalsIgnoreCase(p))
                    {
                        System.out.println(name[i]);
                    }
                }
            }
        }
    }
```

14.
```
class Fibonacci
{
    public static void main(String args[])
    {
        int fib[] = new int[10];
        fib[0] = 0;
        fib[1] = 1;
        for(int i = 2; i < 10; i++)
        {
            fib[i] = fib[i - 1] + fib[i - 2];
        }
        for(int i = 0; i < 10; i++)
        {
            System.out.print(fib[i] + " ");
        }
    }
}
```

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

class EvenOddSort
{
    public static void main(String args[])
    {
```

```
        Scanner sc = new Scanner(System.in);

        int ar[] = new int[10];
        int even[] = new int[10];
        int odd[] = new int[10];
        int e = 0, o = 0;
        for(int i = 0; i < 10; i++)
        {
            ar[i] = sc.nextInt();

            if(ar[i] % 2 == 0)
                even[e++] = ar[i];
            else
                odd[o++] = ar[i];
        }
        for(int i = 0; i < e - 1; i++)
        {
            for(int j = 0; j < e - i - 1; j++)
            {
                if(even[j] > even[j + 1])
                {
                    int t = even[j];
                    even[j] = even[j + 1];
                    even[j + 1] = t;
                }
            }
        }

        System.out.println("Result:”);
        for(int i = 0; i < e; i++)
            System.out.print(even[i] + " ");
        for(int i = 0; i < o; i++)
            System.out.print(odd[i] + " ");
    }
}
```

16.
```
import java.util.*;

class SymmetricMatrix
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);

        System.out.println("Enter size M:”);
        int m = sc.nextInt();

        int a[][] = new int[m][m];
        int flag = 1;

        System.out.println("Enter matrix:”);
        for(int i = 0; i < m; i++)
        {
            for(int j = 0; j < m; j++)
            {
                a[i][j] = sc.nextInt();
            }
        }
        for(int i = 0; i < m; i++)
        {
            for(int j = 0; j < m; j++)
            {
                if(a[i][j] != a[j][i])
                {
                    flag = 0;
                    break;
                }
            }
        }
        if(flag == 1)
            System.out.println("Symmetric Matrix”);
```

```
            else
                System.out.println("Not Symmetric”);
        }
    }
```

17.
```
class PascalTriangle
{
    public static void main(String args[])
    {
        int n = 6;
        int p[][] = new int[n][n];

        for(int i = 0; i < n; i++)
        {
            for(int j = 0; j <= i; j++)
            {
                if(j == 0 || j == i)
                    p[i][j] = 1;
                else
                    p[i][j] = p[i-1][j-1] + p[i-1][j];
                System.out.print(p[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

18.
```
import java.util.*;
class SwapRows
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        int a[][] = new int[3][3];
        System.out.println("Enter matrix:”);
        for(int i = 0; i < 3; i++)
        {
```

```
            for(int j = 0; j < 3; j++)
                a[i][j] = sc.nextInt();
        }
        for(int j = 0; j < 3; j++)
        {
            int temp = a[0][j];
            a[0][j] = a[2][j];
            a[2][j] = temp;
        }
        System.out.println("Result:”);
        for(int i = 0; i < 3; i++)
        {
            for(int j = 0; j < 3; j++)
                System.out.print(a[i][j] + " ");
            System.out.println();
        }
    }
}
```

19.
```
class ColumnMax
{
    public static void main(String args[])
    {
        int a[][] = {
            {5, 7, 4},
            {4, 6, 2},
            {8, 2, 10}
        };
        for(int j = 0; j < 3; j++)
        {
            int max = a[0][j];
            for(int i = 1; i < 3; i++)
            {
                if(a[i][j] > max)
                    max = a[i][j];
            }
```

```
                System.out.println("Highest value in column " + (j+1)
+ " = " + max);
            }
        }
}
```

20.
```
import java.util.*;
class BoundaryMatrix
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        char a[][] = new char[n][n];
        char c1 = sc.next().charAt(0);
        char c2 = sc.next().charAt(0);
        char c3 = sc.next().charAt(0);
        for(int i = 0; i < n; i++)
        {
            for(int j = 0; j < n; j++)
            {
                if(i == 0 || j == 0 || i == n-1 || j == n-1)
                {
                   if((i == 0 || i == n-1) && (j == 0 || j == n-1))
                        a[i][j] = c1;
                    else
                        a[i][j] = c2;
                }
                else
                    a[i][j] = c3;
            }
        }

        for(int i = 0; i < n; i++)
        {
            for(int j = 0; j < n; j++)
```

```
                System.out.print(a[i][j] + " ");
            System.out.println();
        }
    }
}
```

21.
```
import java.util.*;
class Frequency
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);

        System.out.println("Enter size:”);
        int n = sc.nextInt();
        int ar[] = new int[n];
        int visited[] = new int[n];
        System.out.println("Enter elements:”);
        for(int i = 0; i < n; i++)
        {
            ar[i] = sc.nextInt();
            visited[i] = 0;
        }
        System.out.println("Number Frequency”);
        for(int i = 0; i < n; i++)
        {
            if(visited[i] == 1)
                continue;
            int count = 1;
            for(int j = i + 1; j < n; j++)
            {
                if(ar[i] == ar[j])
                {
                    count++;
                    visited[j] = 1;
                }
```

```
                }
                System.out.println(ar[i] + " " + count);
            }
        }
    }
```

22.
```
import java.util.*;

class LowerTriangular
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);

        int n = sc.nextInt();
        int a[][] = new int[n][n];
        int flag = 1;

        System.out.println("Enter matrix:”);

        for(int i = 0; i < n; i++)
        {
            for(int j = 0; j < n; j++)
            {
                a[i][j] = sc.nextInt();
            }
        }

        for(int i = 0; i < n; i++)
        {
            for(int j = 0; j < n; j++)
            {
                if(j > i && a[i][j] != 0)
                {
                    flag = 0;
                }
```

```
                }
            }
            if(flag == 1)
                System.out.println("The Matrix is Lower Triangular”);
            else
              System.out.println("The Matrix is not Lower Triangular”);
        }
    }
```

23.
```
import java.util.*;
class MirrorMatrix
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        int m = 4, n = 4;
        int a[][] = new int[m][n];
        System.out.println("Enter matrix:”);
        for(int i = 0; i < m; i++)
        {
            for(int j = 0; j < n; j++)
            {
                a[i][j] = sc.nextInt();
            }
        }
        System.out.println("Mirror Matrix:”);
        for(int i = 0; i < m; i++)
        {
            for(int j = n - 1; j >= 0; j--)
            {
                System.out.print(a[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

24.
```
import java.util.*;
class MatrixSubtraction
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        int a[][] = new int[3][3];
        int b[][] = new int[3][3];
        int c[][] = new int[3][3];
        System.out.println("Enter matrix A:”);
        for(int i = 0; i < 3; i++)
            for(int j = 0; j < 3; j++)
                a[i][j] = sc.nextInt();
        System.out.println("Enter matrix B:”);
        for(int i = 0; i < 3; i++)
            for(int j = 0; j < 3; j++)
                b[i][j] = sc.nextInt();
        for(int i = 0; i < 3; i++)
        {
            for(int j = 0; j < 3; j++)
            {
                c[i][j] = a[i][j] - b[i][j];
            }
        }
        System.out.println("Result Matrix:”);
        for(int i = 0; i < 3; i++)
        {
            for(int j = 0; j < 3; j++)
            {
                System.out.print(c[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

25.
```
import java.util.*;

class Array4x3
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        int a[][] = new int[4][3];
        System.out.println("Enter elements:”);
        for(int i = 0; i < 4; i++)
        {
            for(int j = 0; j < 3; j++)
            {
                a[i][j] = sc.nextInt();
            }
        }
        int sumOddPosOdd = 0;
        System.out.println("Non-boundary elements:”);
        for(int i = 0; i < 4; i++)
        {
            for(int j = 0; j < 3; j++)
            {
                if(i != 0 && j != 0 && i != 3 && j != 2)
                    System.out.print(a[i][j] + " ");

                if((i + j) % 2 != 0 && a[i][j] % 2 != 0)
                    sumOddPosOdd += a[i][j];
            }
        }

        System.out.println("\nSum of odd numbers in odd positions:
" + sumOddPosOdd);
    }
}
```

26.
```
import java.util.*;
class MeritList
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        String name[] = new String[40];
        int marks[] = new int[40];

        for(int i = 0; i < 40; i++)
        {
            name[i] = sc.next();
            marks[i] = sc.nextInt();
        }
        for(int i = 0; i < 39; i++)
        {
            for(int j = 0; j < 39 - i; j++)
            {
                if(marks[j] < marks[j+1])
                {
                    int t = marks[j];
                    marks[j] = marks[j+1];
                    marks[j+1] = t;
                    String s = name[j];
                    name[j] = name[j+1];
                    name[j+1] = s;
                }
            }
        }
        for(int i = 0; i < 40; i++)
        {
            System.out.println(name[i] + " " + marks[i]);
        }
    }
}
```

27.
```
class MatrixMultiply
{
    public static void main(String args[])
    {
        int A[][] = {
            {3, 2, 1, 2},
            {6, 4, 5, 0},
            {7, -1, 0, 2},
            {4, 3, 1, 1}
        };
        int B[][] = {
            {3, 6, -5, 2},
            {5, 3, 4, 6},
            {-17, -38, 2, -8},
            {0, -2, -2, 5}
        };
        int C[][] = new int[4][4];
        for(int i = 0; i < 4; i++)
        {
            for(int j = 0; j < 4; j++)
            {
                C[i][j] = 0;

                for(int k = 0; k < 4; k++)
                {
                    C[i][j] += A[i][k] * B[k][j];
                }
            }
        }
        for(int i = 0; i < 4; i++)
        {
            for(int j = 0; j < 4; j++)
            {
                System.out.print(C[i][j] + " ");
            }
```

```
                System.out.println();
            }
        }
    }
```

28.
```
import java.util.*;
class Denomination
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        int amt = sc.nextInt();
        if(amt <= 0 || amt > 99999)
        {
            System.out.println("Invalid Amount”);
            return;
        }
        int temp = amt;
        String word = "”;

        while(temp > 0)
        {
            int d = temp % 10;
            if(d == 0) word = "Zero " + word;
            else if(d == 1) word = "One " + word;
            else if(d == 2) word = "Two " + word;
            else if(d == 3) word = "Three " + word;
            else if(d == 4) word = "Four " + word;
            else if(d == 5) word = "Five " + word;
            else if(d == 6) word = "Six " + word;
            else if(d == 7) word = "Seven " + word;
            else if(d == 8) word = "Eight " + word;
            else if(d == 9) word = "Nine " + word;
            temp /= 10;
        }
```

```
            System.out.println(word);
            int n = amt;
            int notes[] = {500, 200, 100, 50, 20, 10, 1};
            for(int i = 0; i < notes.length; i++)
            {
                int count = n / notes[i];
                if(count != 0)
                {
                    System.out.println(notes[i] + " * " + count + " = "
+ (notes[i]*count));
                    n = n % notes[i];
                }
            }
        }
}
```

12.1 Installation & IDE and Fundamentals of Python Programming

MIND DRILL

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

B. 1. interpreter 2. virtual 3. simple syntax 4. user-defined 5. literal

C. 1. Portability and compatibility mean Python programs can run on different computer systems and operating systems without significant changes to the code.

2. Python is a dynamically typed programming language. This means you do not need to declare the data type of a variable before using it. Python automatically understands the type of a variable when the program runs. The variable name acts as a reference to the value stored in memory and the data type is automatically decided at run-time based on the assigned value. This makes coding easier and faster.

3. Keywords, also known as reserved words, are words that have a fixed meaning and specific use in a programming language. The language defines their purpose, so they can't be used for naming variables or functions. Every programming language has its own set of keywords.

4. There are different types of tokens in Python:
   - Keywords: Keywords, also known as reserved words, are words that have a fixed meaning and specific use in a programming language.
   - Identifiers: An identifier is the name given to a variable, function, class, object or any other element in a Python program.
   - Literals: In Python, literals are fixed or constant values that are written directly in a program.
   - Operators: In Python, operators are special symbols that perform specific operations on variables and values.
   - Punctuators (Separators): In Python, punctuators (or delimiters) are symbols that help organize and structure the code.
5. Here are some uses of multi-line comments:
   - To explain large blocks of code
   - To write detailed program descriptions
   - To improve code readability
   - Useful for documentation purposes
6. Integrated Development Learning Environment (IDLE) is the default IDE that comes automatically when Python is installed on a computer. It provides a simple and easy-to-use environment for writing and running Python programs. You can interact with Python IDLE in two different modes:
   - Interactive Mode (Python IDLE-Shell Mode): In this mode, you type one command at a time in front of the Python command prompt (>>>) and Python gives you the result based on the command given.
   - Script Mode (Editor): In this mode, you save all your commands together in the text editor with the extension .py. When you run this text file .py then the output of the commands will be displayed in Shell mode.
7. In Python, literals are fixed or constant values that are written directly in a program. They represent data that does not change during the execution of the program. The different types of literals in Python are given below:

   Numeric literals: x = 10
   y = 3.14
   z = 5+2j

   String literals: name = "Python"
   msg = 'Hello'

   Boolean literals: a = True
   b = False

   Special literal: x = None
8. In Python, the terms L-value and R-value are used to describe how variables and values are used in an assignment statement:

- L-value (Left value): The L-value is the variable that appears on the left side of the assignment operator (=). It represents a memory location where a value will be stored. In simple words, it is the name that receives the value.
- R-value (Right value): The R-value is the value or expression that appears on the right side of the assignment operator (=). It provides the data that will be assigned to the variable.

For example,

x = 10

x → L-value (the variable that stores the value)

10 → R-value (the value being assigned)

9. In Python, punctuators (or delimiters) are symbols that help organize and structure the code. They do not perform any calculations or comparisons like operators do, but they make the program easier to read and understand by separating statements, grouping code and showing the proper structure. The different types of punctuators in Python is shown in Table:

Punctuators	Description
()	Used in function calls
[]	Used in lists
{}	Used in dictionaries
,	Separates items
:	Used in blocks and dictionaries
.	Access object members

10. PyCharm is a powerful IDE developed specifically for Python programming. It provides advanced tools such as intelligent code completion, error detection, debugging and project management which help developers write and manage Python programs more efficiently.

D. 1. In Python, double quotes (" ") are used to create strings and display messages using the print() function. Triple quotes (""" """) are used for multi-line strings and docstrings that explain programs, functions, or classes. Use double quotes for user messages and triple quotes for documentation and long text.

2. Debugging and testing features in an IDE help a student find and fix errors in a Python program in the following way:
   - Debugging allows the programmer to run the program step by step, check the values of variables, and identify the exact line where the error occurs.
   - Testing helps verify whether the program produces the correct results for different inputs after the errors are fixed.

E. 1. Punctuators are important symbols that help structure Python code correctly. Mistakes in using them cause errors.

2. The first student forgot commas between the list items.

3. The second student misused the print format.

4. The third student skipped the curly braces while creating the dictionary.

F. 1. Program:

```
# Store personal details in variables
name = "Pooja”
age = 12
student_class = "XI”
# Display the details
print("Name:”, name)
print("Age:”, age)
print("Class:”, student_class)
Output
Name: Pooja
Age: 12
Class: XI
```

2. Program:

```
# Program to calculate area and perimeter of a rectangle
length = 12
breadth = 8
area = length * breadth
perimeter = 2 * (length + breadth)
print("Length =”, length)
print("Breadth =”, breadth)
print("Area =”, area)
print("Perimeter =”, perimeter)
Output
Length = 12
Breadth = 8
Area = 96
Perimeter = 40
```

3. Program:

```
# Program to calculate the sum of two numbers
# Step 1: Take input of first number from the user
num1 = float(input("Enter first number: "))
# Step 2: Take input of second number from the user
num2 = float(input("Enter second number: "))
# Step 3: Add both numbers
```

```
sum_result = num1 + num2
# Step 4: Display the result
print("The sum of the two numbers is:", sum_result)
Output
Enter first number: 12
Enter second number: 8
The sum of the two numbers is: 20.0
```

G. Find the output of the following programs:

1. 24
2. The code gives the output: class = 12
   ^
   SyntaxError: invalid syntax
3. ['Apple', 'Mango', 'Banana']
   ('Red', 'Blue', 'Pink', 'Orange')

12.2 Data Types in Python

MIND DRILL

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

B. 1. decimal 2. single 3. Boolean 4. key-value 5. modification

C. 1. Support type conversion means that numeric values can be converted from one type to another using functions like int(), float() and complex().

2. The two characteristics of Set data types in Python:
   - Collection of Distinct Elements: A set automatically removes duplicate values and stores only unique items.
   - Unordered Structure: Elements are not stored in a fixed position and the order may change.
3. In Boolean data types, automatically evaluated means many values are automatically interpreted as Boolean in conditions (e.g., 0 is treated as False, non-zero numbers as True).
4. A tuple is a sequence data type in Python that stores multiple values in an ordered form. The elements of a tuple are enclosed in parentheses () and separated by commas. Each element has an index number starting from 0 and a tuple is immutable, which means its elements cannot be changed after creation.

5. Mutable data types are types of objects in Python whose values can be changed after they are created. You can modify, add or remove elements without creating a new object in memory. Mutable objects can be homogeneous, meaning they can contain elements of the same data type. Examples of mutable data types include sets.

   An immutable object in Python is an object whose value cannot be changed after it is created. Once you assign a value to it, you cannot modify the same object. If you try to change it, Python creates a new object instead of modifying the original one.

6. The main characteristics of None data types in Python:
   - Represents Absence of Value: None is used to indicate that a variable has no value or no data.
   - Single Constant Object: There is only one None object in Python.
   - Belongs to NoneType: The data type of None is NoneType.
   - Immutable: The value None cannot be changed once assigned.
   - Common Default Return Value: Functions that do not return any value automatically return None.
   - Used as Placeholder: Often used to initialize variables when the actual value is not yet known.
   - Case-Sensitive Keyword: None must always begin with a capital "N".
7. Some different ways to use a list are given below:
   - Basic Lists: A basic list is a list that stores elements of the same data type. The elements are written inside square brackets [] and separated by commas.
   - Mixed List: A mixed list is a list that stores different types of data in one list. Its elements do not need to be of the same data type and can include integers, strings, floats and Boolean values together.
   - Nested List: A nested list is a list that contains one or more lists as its elements. It means a list inside another list, used to store data in a structured or tabular form.
8. A string is a sequence data type in Python that stores a group of characters enclosed in single (' ') or double (" ") quotes. A string in Python is an immutable sequence of characters enclosed in quotes. It is used to represent text such as names, messages, sentences or any combination of letters, numbers and symbols.

D. 1. Python lists can store different data types together by using structures like dictionaries or tuples inside a list. Each book's details such as ID, title, author, price, genre and format can be stored as a dictionary, allowing organised storage of mixed information within one list.

2. Unique employee IDs are important because dictionary keys must be distinct. If duplicate IDs exist, later values overwrite earlier ones, causing data loss. This reduces data integrity and leads to incorrect or missing records. Unique IDs ensure accurate identification, retrieval and management of each employee's information.

E. 1. A list helps in managing student data by storing multiple students' details in a single collection, where each student's information is kept in a sublist.

2. A student's data is structured as a sublist containing student name, list of courses, and number of courses, and all students are stored inside a main list.

3. It is important because mutability allows the system to update student information such as adding or removing courses without creating a new list.
4. The system can track the number of courses by storing the length of the courses list for each student as a separate value in the student's sublist.

F. 1. Program:

```
# Program to display five integer numbers in a list
numbers = [10, 20, 30, 40, 50]
print("The list of integer numbers is:")
print(numbers)
```

Output

```
The list of integer numbers is:
[10, 20, 30, 40, 50]
```

2. Program:

```
# Python program to create a tuple of four subjects and print the first subject using indexing
# Creating a tuple of four subjects
subjects = ("Maths", "Science", "English", "Computer")
# Printing the first subject using indexing
print("First subject is:", subjects[0])
```

Output:

```
First subject is: Maths
```

3. Program:

```
# Python program to create a dictionary and print the salary value
employee = {
    "id": 101,
    "salary": 50000
}
print("Salary:", employee["salary"])
```

Output

```
Salary: 50000
```

G. 1. Country and City Name: India Jaipur

2. Student Data: [['Amit', 85], ['Riya', 90], ['Karan', 78]]
   First Student: ['Amit', 85]
3. Colors: {'Blue', 'Yellow', 'Red', 'Green'}
   Numbers: {40, 10, 20, 30}

12.3 Data Processing in Python

MIND DRILL

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

B. 1. string 2. integer 3. ZeroDivisionError 4. try-except
5. crash

C. 1. This text-based interface, known as a Command Line Interface (CLI), is a powerful tool for building applications, from simple scripts to complex software.

2. The two Python functions essential for console I/O (Input/Output) are:
   - input(): Used to take input from the user.
   - print(): Used to display output on the screen.

3. The input() function can accept an optional string argument, known as a prompt. It helps the user understand what type of information they should enter. By default, any value entered by the user using the input() function is of string data type.

4. Errors in Python are broadly classified into three types:
   - Syntax errors: Syntax errors, also known as parsing errors, are the most basic kind of error.
   - Run-time errors: Run-time errors, also known as exceptions, occur while the program is running.
   - Logical errors: A logical error is a mistake in the design or thinking behind a program.

5. Syntax errors, also known as parsing errors, are the most basic kind of error. They occur when the Python interpreter encounters code that violates the grammatical rules of the Python language. The interpreter cannot understand your instructions, so the program fails before it even starts running.

6. A logical error is a mistake in the design or thinking behind a program. In this type of error, the program runs without showing any syntax or runtime error, but the output produced is incorrect. This happens because the steps written in the program do not correctly represent the intended logic of the problem. Logical errors are often difficult to identify because the program appears to work normally. The code executes successfully, but due to incorrect formulas, wrong conditions, or incorrect sequence of steps, the result does not match what the programmer expected.

7. Run-time errors, also known as exceptions, occur while the program is running. The syntax of the code is correct, so the program starts, but during execution, something unexpected happens that forces the program to stop. These errors are often caused by user input or unforeseen states.

   some common types of run-time errors with their examples are as follows:
   - ValueError: Occurs when a function receives an argument of the correct type but an inappropriate value. For example, int("abc").

- NameError: Occurs when you try to use a variable or function that has not been defined. For example, print(age) (if age was never assigned a value).
- TypeError: Occurs when you try to perform an operation on a data type that doesn't support it. For example, "hello" / 5.
- ZeroDivisionError: Occurs when you try to divide a number by zero. For example, result = 10 / 0.
- IndexError: Occurs when you try to access an element from a list or tuple using an index that is out of range. For example, my_list = [1,2,3]; print(my_list[5])

8. When you use the input() function to enter numbers, Python treats the value as a string. It can not be used for mathematical calculations. If you try to add two numbers entered by the user, Python will join the strings together instead of performing actual addition.

D. 1. The logical error is the incorrect formula used for compound interest. The program multiplies the amount by time instead of raising it to the power of time and subtracting the principal. The correct formula is: CI = P * (1 + R/100) ** T - P, giving the expected interest of 2762.81.

2. A NameError occurs when a program uses a variable that has not been defined. For example, if username is printed before it receives user input, the program will fail. To avoid this error, declare and assign all variables before using them and check variable names for correct spelling.

E. 1. The logical error is that subject5 is not included while calculating total_marks, even though the total is divided by 5 to find the average.

2. The program runs without errors because the Python syntax is correct. However, the calculation is incorrect since the marks of the fifth subject are not added, resulting in a logical error.

3. The error can be corrected by including subject5 in the total calculation: total_marks = subject1 + subject2 + subject3 + subject4 + subject5, and then dividing by 5 to get the correct average.

F. 1. Program:

```
# Program to find the length of a string
text = input("Enter a string: ")
print("Length of the string:", len(text))
Output
Enter a string: Computer science
Length of the string: 17
```

2. Program:

```
# Program to find the sum of digits in a number
num = int(input("Enter a number: "))
sum_digits = 0
while num > 0:
    digit = num % 10
    sum_digits += digit
```

```
        num = num // 10
    print("Sum of digits:", sum_digits)
    Output
    Enter a number: 56
    Sum of digits: 11
```

3. Program:

```
# Program to display common elements between two lists
list1 = [10, 20, 30, 40, 50]
list2 = [30, 40, 50, 60, 70]
common = []
for item in list1:
    if item in list2:
        common.append(item)
print("Common elements:", common)
Output
Common elements: [30, 40, 50]
```

G. 1. Output:

```
Enter your school name: Dehradun Public School
My school name is Dehradun Public School
```

2. Output:

```
Enter first value: 6
Enter second value: 8
Values | 6 | 8
```

3. Output:

```
The sum of 85 and 90 is 175.
```

12.4 Operators and Expressions

MIND DRILL

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

B. 1. Assignment 2. operands 3. not found 4. Left, Right 5. logical 6. identity

C. 1. The not operator in Python is a logical operator used to reverse the truth value of a condition. If a condition is True, the not operator changes it to False, and if the condition is False, it changes it to True. It is commonly used in conditional statements such as if statements to check whether a condition is not satisfied.

2. An operator is a special symbol or keyword in Python that tells the computer to perform a specific operation on one or more values, called operands. Operators are used to carry out tasks such as mathematical calculations, comparisons, logical decisions, assignments and data manipulation.

3. A unary operator in Python is a type of operator that works on a single operand to perform an operation. Common unary operators include + (positive), - (negation) and not (logical NOT).

4. The difference between implicit and explicit type conversion in Python are as follows:

   - Implicit Conversion is the automatic conversion of one data type into another data type by Python. It is also known as automatic type conversion. When two different data types are used in an expression, Python automatically converts the smaller data type into the larger data type to avoid data loss.
   - Explicit conversion is the process of manually converting one data type into another data type in a program using built-in functions. In this method, the programmer clearly specifies the desired data type to which a value should be converted.

5. A unary operator in Python is a type of operator that works on a single operand to perform an operation. Common unary operators include + (positive), - (negation) and not (logical NOT). They are used to change the sign of a number, reverse a Boolean value or perform an operation on a single value, without needing a second operand.

   For example:

   ```
   x = 10
   print(-x)
   ```

   A binary operator in Python is a type of operator that works on two operands to perform an operation. Common binary operators include + (addition), - (subtraction), * (multiplication), / (division) and (logical AND) or (logical OR), == (equal to) and > (greater than). They are used to perform mathematical calculations, comparisons, logical operations or bitwise manipulations between two values.

   For example:

   ```
   a = 10
   b = 5
   print(a + b)   # Output: 15
   ```

   A ternary operator in Python is a type of operator that works on three operands to perform a conditional operation. It allows selecting a value based on a condition in a single line, the ternary operator allows you to choose between two values based on a condition using only one line of code.

   For example:

   ```
   age = 18
   ```

result = "Eligible to vote" if age >= 18 else "Not eligible to vote"

print(result)

6. The following table shows some bitwise operators with examples:

Operator	Description	Examples

& (Bitwise AND)	Returns 1 in each bit position if both corresponding bits are 1.	a = 5 # Binary: 0101
		b = 3 # Binary: 0011
		print(a & b)

| (Bitwise OR)	Returns 1 in each bit position if at least one corresponding bit is 1.	a = 5 # Binary: 0101
		b = 3 # Binary: 0011
		print(a | b)

^ (Bitwise XOR)	Returns 1 in each bit position if the corresponding bits are different.	a = 5 # Binary: 0101
		b = 3 # Binary: 0011
		print(a ^ b)

7. Identity operators are used to determine whether two variables point to the same object in memory. They do not compare the actual values of the objects; instead, they check if both variables refer to the exact same location. The result of an identity operation is always either True or False.

8. Logical operators are commonly used when a program needs to check multiple conditions at the same time or control the flow of execution. The following table shows the logical operators in Python along with their description:

Operator	Description
not	Returns True if operand is false.
and	Returns True if both operands are true.
or	Returns True if at least one operand is true.

D. 1. For year = 1900, the correct leap year rule requires checking divisible by 400 OR not divisible by 100. Since 1900 is divisible by 100 but not by 400, it is not a leap year. After mistakenly replacing the condition, the logic still evaluates as false for leap year, so the output will be "Not a Leap Year".

2. Initially, a = 10 and b = 5. If the correct operator a += b is used, result is 15. But with the mistaken operator a -= b, the operation becomes 10 − 5, so the final value of a becomes 5.

E. 1. a. A Python program for Bitwise AND (&).

Program:

```
a = 12
b = 5
print(a & b)
```

Output

4

b. A Python program for Bitwise OR (|).

Program:

```
a = 12
b = 5
print(a | b)
```

Output

13

2. Given:

x = 4, y = 3

Step 1: Convert to binary (4-bit form)

4 = 0100

3 = 0011

Step 2: Apply Bitwise XOR (^) rule

(XOR gives 1 only when bits are different)

```
 0100
^ 0011
------
 0111
```

Step 3: Convert result back to decimal

0111 = 7

Final Answer:

x ^ y = 7

3. When the Bitwise NOT (~) operator is applied to 5, it inverts all its bits. Binary: 5 = 00000101, ~5 = 11111010. In Python, the result is -6 (~5 = -(5 + 1)).

4. Program:

```
# Given number
num = 5
# Left Shift by 2
left_shift = num << 2
# Right Shift by 2
right_shift = num >> 2
# Display results
print("Original Number:", num)
print("Binary of", num, ":", format(num, '08b'))
print("\nLeft Shift (<< 2):")
```

```
print("Binary:", format(num, '08b'), "<< 2 =", format(left_shift, '08b'))
print("Decimal Result:", left_shift)
print("\nRight Shift (>> 2):")
print("Binary:", format(num, '08b'), ">> 2 =", format(right_shift, '08b'))
print("Decimal Result:", right_shift)
```

Output

```
Original Number: 5
Binary of 5 : 00000101
Left Shift (<< 2):
Binary: 00000101 << 2 = 00010100
Decimal Result: 20
Right Shift (>> 2):
Binary: 00000101 >> 2 = 00000001
Decimal Result: 1
```

F. 1. Program:

```
# Program to calculate the remainder
num1 = int(input("Enter the first number: "))
num2 = int(input("Enter the second number: "))
remainder = num1 % num2
print("Remainder:", remainder)
```

Output

```
Enter the first number: 15
Enter the second number: 6
Remainder: 3
```

2. Program:

```
# Program to check whether a number is even or odd
num = int(input("Enter a number: "))
if num % 2 == 0:
    print("The number is Even.")
else:
    print("The number is Odd.")
```

Output

```
Enter a number: 56
The number is Even.
```

3. Program:

```
# Program to perform left shift operation
num = int(input("Enter an integer: "))
shift = int(input("Enter the number of positions to shift: "))
result = num << shift
print("Result after left shift:", result)
```

Output

Enter an integer: 5

Enter the number of positions to shift: 2

Result after left shift: 20

G. 1. Output:

Floor Division: 3

Modulus: 3

2. Output:

Exponentiation Result: 512

3. Output:

Logical Operation Result: True

12.5 Flow of Control

MIND DRILL

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

B. 1. placeholder 2. range() 3. updation 4. unfixed/indefinite 5. outer

C. 1. The continue statement is used to skip the current iteration of the loop and jump to the next iteration. When continue is executed, the rest of the code inside the loop for that particular iteration is skipped and the loop moves on to the next iteration.

2. The control variable in a loop keeps track of the loop's progress and determines how many times the loop executes. It is updated after each iteration until the loop condition becomes false.

3. In Python, repetition is done using loop statements:

   - The for loop is a type of iterative statement in Python that allows you to repeat a block of code to a specific number of times or iterate over a sequence (like a list, tuple, string or range).

- The while loop in Python is a type of iterative statement that is used to repeat a block of code as long as a specified condition is true.

4. The nested if statement in Python is a way of placing one if statement inside another. This allows you to check multiple conditions in a hierarchical manner. It is useful when you need to make decisions based on the outcome of previous conditions. A nested if statement is used when you want to check a condition inside another condition. This makes your code more flexible for complex decision-making scenarios.
5. Program:

```
# Program to check divisibility by 2 and 3
num = int(input("Enter a number: "))
if num % 2 == 0 and num % 3 == 0:
    print("The number is divisible by both 2 and 3.”)
elif num % 2 == 0:
    print("The number is divisible by 2 only.”)
elif num % 3 == 0:
    print("The number is divisible by 3 only.”)
else:
    print("The number is divisible by neither 2 nor 3.”)
```

Output:

Enter a number: 88

The number is divisible by 2 only.

6. A for loop in Python is used to repeat a block of code a fixed number of times. It commonly works with the range() function, which generates a sequence of numbers. The loop variable takes each value in the sequence, executes the loop body, and stops automatically when the sequence ends.
7. Conditional and iterative flow of control make Python programs dynamic and flexible by allowing them to make decisions and repeat tasks automatically. Conditional statements execute different actions based on conditions, while iterative statements repeatedly execute code until a condition is met or a sequence ends, reducing repetition and improving program efficiency.
8. Program

```
# Python program to check whether numbers are even or odd
for num in range(1, 11):
    if num % 2 == 0:
        print(num, "is Even”)
    else:
        print(num, "is Odd”)
```

Output

1 is Odd

2 is Even

3 is Odd

4 is Even

5 is Odd

6 is Even

7 is Odd

8 is Even

9 is Odd

10 is Even

In the program, the for loop iterates through numbers from 1 to 10 using the range() function. During each iteration, the if-else conditional statement checks whether the current number is divisible by 2. The flow of control is managed by the loop and conditional statement, making the program dynamic and flexible.

D. 1. A nested if statement should be used. It first checks whether the student has passed. If the student has passed, another if statement checks whether the student has scored enough marks to receive a distinction.

2. The program should combine these three types of flow of control:
   - Sequential flow of control: To execute statements in order.
   - Conditional flow of control: To make decisions using if, elif, and else.
   - Iterative flow of control: To repeat tasks using loops such as for and while.

E. 1. Since 8°C is below 10°C, the system will advise people to wear heavy winter clothes.

2. The elif statement checks each temperature range one by one after the if condition. When the correct temperature range is found, it executes the corresponding advice and skips the remaining conditions, ensuring users receive the appropriate weather advice.

F. 1.
```
for i in range(1, 5):
    print("*" * i)
```

2.
```
for i in range(1, 9):
    if i % 2 == 0:
        print(i)
```

3.
```
print("Fizz")
print("Buzz")
print("Fizz")
```

G. 1. Output:

0 0

0 1

0 2
1 0
1 1
1 2

2. Output:
   1 Odd
   2 Even
   3 Odd
   4 Even
   5 Odd
3. Output:
   PYTH

13. Trends in Computing and Ethical Issues

MIND DRILL

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

B. 1. viruses 2. Spam 3. learn 4. Trojan Horse
5. Privacy breach

C. 1. Phishing is a cybercrime where fake emails/messages are sent to trick users into revealing sensitive information like passwords, bank details or OTPs. It works by pretending to be a trusted source such as a bank or company.

2. AI helps in smart cities by improving traffic management, healthcare services, waste management, smart lighting and other automated city systems for better efficiency.
3. IoT in healthcare is used to monitor patient health using smart devices like fitness trackers, heart rate monitors and smartwatches, and also to send alerts in emergencies.
4. Intellectual property rights are legal rights that protect original creations like inventions, software, music, designs and brand names from unauthorised use or copying.
5. VR in medical training helps students practise surgeries and medical procedures in a safe virtual environment without risk to real patients.
6. IoT in smart homes is used to connect devices like smart TVs, refrigerators and lights so they can be controlled remotely, automate tasks and improve convenience and energy use.

7. Ethical implications of AR in education include privacy concerns, distraction risks, over-dependence on technology and ensuring accurate and appropriate digital content in learning.
8. Phishing uses social engineering by pretending to be a trusted person or organisation and manipulating users emotionally (fear, urgency or reward) to make them share personal information.
9. Digital arrest scams can cause huge financial losses, often draining victims' savings and assets. Victims may transfer large amounts of money under fear and pressure, leading to financial distress, debt, and loss of economic stability, especially among elderly people.
10. Businesses ensure compliance by using licensed software only, respecting copyrights and patents, avoiding piracy, properly registering their intellectual property, and educating employees about legal use of digital content and software.

D. 1. AI and ML can detect scams by:

- Analysing unusual call patterns, messages, and transactions
- Identifying fraud keywords and fake identities in real time
- Detecting suspicious bank transfers or OTP activity
- Blocking known scam numbers using predictive models
- Example: Banks use AI to flag sudden large transfers during a call and freeze transactions instantly.

2. **Opportunities:**

•Makes learning interactive and immersive

•Helps students explore historical places and geographic locations virtually

•Improves understanding through 3D visualisation and simulations

Drawbacks:

•High cost of VR devices limits accessibility

•May cause motion sickness or discomfort

•Can reduce real-world interaction and attention span

•Requires strong internet and technical setup

E. 1. a. 2. a. 3. d