Ex.No.1	**Develop programs using flow control statements and arrays to manage execution flow and data organization effectively.**

1.a) Student Grades Evaluation System Write a Java program that takes an array of student grades as input. Use control statements to classify each grade (e.g., A, B, C, D, F) and display the number of students in each category. generate aim, algorithm, program and output in java code

Aim:

To write a Java program that reads an array of student grades (percentages), classifies them into letter grades (A, B, C, D, F), and displays the number of students in each category using control statements.

Algorithm:

1. Start the program.
2. Define an array of student grades (or input them).
3. Initialize counters for each grade category (A, B, C, D, F).
4. Loop through each grade:
    - Use if-else or switch to classify:
        - A: 90–100
        - B: 80–89
        - C: 70–79
        - D: 60–69
        - F: below 60
    - Increment the corresponding counter.
5. Display the count of students in each category.
6. End the program.

Program:

```
public class StudentGradesEvaluation {
   public static void main(String[] args) {
      // Sample array of student grades (in percentage)
      int[] grades = {95, 82, 76, 89, 67, 54, 100, 73, 85, 61, 49, 90};

      // Counters for grade categories
      int countA = 0, countB = 0, countC = 0, countD = 0, countF = 0;

      // Classify each grade
      for (int grade : grades) {
         if (grade >= 90 && grade <= 100) {
            countA++;
         } else if (grade >= 80 && grade < 90) {
            countB++;
         } else if (grade >= 70 && grade < 80) {
            countC++;
         } else if (grade >= 60 && grade < 70) {
            countD++;
         } else {
            countF++;
         }
      }
```

```
        // Display results
        System.out.println("Student Grades Classification:");
        System.out.println("Number of students with grade A (90-100): " + countA);
        System.out.println("Number of students with grade B (80-89): " + countB);
        System.out.println("Number of students with grade C (70-79): " + countC);
        System.out.println("Number of students with grade D (60-69): " + countD);
        System.out.println("Number of students with grade F (0-59): " + countF);
    }
}
```

Output:

Student Grades Classification:
Number of students with grade A (90-100): 3
Number of students with grade B (80-89): 3
Number of students with grade C (70-79): 2
Number of students with grade D (60-69): 2
Number of students with grade F (0-59): 2

1.b) Array Maximum and Minimum Finder

Develop a Java program that reads integers into an array and uses loop statements to determine and display the maximum and minimum values from the array.

Aim:

To write a Java program that reads integers into an array and uses loop statements to find and display the maximum and minimum values in the array.

Algorithm:

1. Start the program.
2. Define or read an array of integers.
3. Initialize max and min with the first element of the array.
4. Loop through the array starting from the second element:
    - If the current element is greater than max, update max.
    - If the current element is less than min, update min.
5. After the loop, display max and min.
6. End the program.

Program:

```
import java.util.Scanner;

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

        // Input size of array
        System.out.print("Enter number of elements: ");
        int n = scanner.nextInt();

        // Input array elements
        int[] numbers = new int[n];
        System.out.println("Enter " + n + " integers:");
        for (int i = 0; i < n; i++) {
            numbers[i] = scanner.nextInt();
        }
```

```
        // Initialize max and min
        int max = numbers[0];
        int min = numbers[0];

        // Loop to find max and min
        for (int i = 1; i < n; i++) {
            if (numbers[i] > max) {
                max = numbers[i];
            }
            if (numbers[i] < min) {
                min = numbers[i];
            }
        }

        // Display results
        System.out.println("Maximum value in array: " + max);
        System.out.println("Minimum value in array: " + min);

        scanner.close();
    }
}
```

Output:

```
Enter number of elements: 5
Enter 5 integers:
23
15
89
4
67
Maximum value in array: 89
Minimum value in array: 4
```

1.c) Even and Odd Counter

Write a Java program that accepts an array of integers. Use a loop to count and print the number of even and odd numbers using if statements.

Aim:

To develop a Java program that reads integers into an array, uses a loop to count how many numbers are even and how many are odd using if statements, and prints the results.

Algorithm:

1. Start the program.
2. Read the number of elements and create an array of that size.
3. Use a loop to read all elements into the array.
4. Initialize two counters: evenCount and oddCount.
5. Loop through the array:
    - If the element is divisible by 2 (num % 2 == 0), increment evenCount.
    - Else, increment oddCount.
6. Print the total number of even and odd numbers.
7. End the program.

Program:

```
import java.util.Scanner;

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

      // Input size of array
      System.out.print("Enter the number of elements: ");
      int n = scanner.nextInt();

      int[] numbers = new int[n];

      // Input elements into array
      System.out.println("Enter " + n + " integers:");
      for (int i = 0; i < n; i++) {
         numbers[i] = scanner.nextInt();
      }

      // Initialize counters
      int evenCount = 0;
      int oddCount = 0;

      // Count even and odd numbers
      for (int i = 0; i < n; i++) {
         if (numbers[i] % 2 == 0) {
            evenCount++;
         } else {
            oddCount++;
         }
      }

      // Output results
      System.out.println("Number of even numbers: " + evenCount);
      System.out.println("Number of odd numbers: " + oddCount);

      scanner.close();
   }
}
```

Output:

```
Enter the number of elements: 6
Enter 6 integers:
12
7
9
24
8
3
Number of even numbers: 3
Number of odd numbers: 3
```

1.d) Linear Search Implementation

Write a Java program that implements a linear search algorithm using an array. Prompt the user to enter the search key and use a loop and conditional logic to determine whether the element is found.

Aim:
To write a Java program that performs a **linear search** on an array using a loop and conditional logic to determine whether a user-provided key is present in the array.

Algorithm:
1. Start the program.
2. Ask the user for the number of elements and create an array of that size.
3. Prompt the user to enter the array elements.
4. Prompt the user to enter the search key.
5. Use a loop to go through each element of the array:
    - If the current element matches the search key, display the index and mark it as found.
6. If the key is not found after the loop, display a message indicating that.
7. End the program.

Program:
```
import java.util.Scanner;

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

      // Input number of elements
      System.out.print("Enter the number of elements: ");
      int n = scanner.nextInt();

      int[] array = new int[n];

      // Input elements
      System.out.println("Enter " + n + " integers:");
      for (int i = 0; i < n; i++) {
         array[i] = scanner.nextInt();
      }

      // Input search key
      System.out.print("Enter the search key: ");
      int key = scanner.nextInt();

      // Linear search logic
      boolean found = false;
      for (int i = 0; i < n; i++) {
         if (array[i] == key) {
            System.out.println("Element " + key + " found at index: " + i);
            found = true;
            break;
         }
      }

      if (!found) {
         System.out.println("Element " + key + " not found in the array.");
      }
```

```
        scanner.close();
    }
}
```
Output:
Enter the number of elements: 5
Enter 5 integers:
10
25
30
45
50
Enter the search key: 30
Element 30 found at index: 2

1.e) Sort Numbers in Ascending Order

Create a Java program that accepts an array of integers and sorts them in ascending order using nested for loops (Bubble Sort). Display the sorted array.

Aim:

To develop a Java program that accepts an array of integers from the user and sorts them in ascending order using **nested for loops** (Bubble Sort algorithm). The program then displays the sorted array.

Algorithm:

1. Start the program.
2. Prompt the user to enter the number of elements.
3. Read elements into an integer array.
4. Use nested for loops to implement the Bubble Sort algorithm:
    - For each element, compare it with the next.
    - Swap if the current element is greater than the next.
5. After sorting, print the sorted array.
6. End the program.

Program:

```
import java.util.Scanner;

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

        // Step 1: Read array size
        System.out.print("Enter the number of elements: ");
        int n = scanner.nextInt();

        int[] numbers = new int[n];

        // Step 2: Read array elements
        System.out.println("Enter " + n + " integers:");
        for (int i = 0; i < n; i++) {
            numbers[i] = scanner.nextInt();
        }

        // Step 3: Bubble Sort algorithm
        for (int i = 0; i < n - 1; i++) {
```

```
            for (int j = 0; j < n - i - 1; j++) {
                if (numbers[j] > numbers[j + 1]) {
                    // Swap elements
                    int temp = numbers[j];
                    numbers[j] = numbers[j + 1];
                    numbers[j + 1] = temp;
                }
            }
        }

        // Step 4: Display sorted array
        System.out.println("Sorted array in ascending order:");
        for (int num : numbers) {
            System.out.print(num + " ");
        }

        scanner.close();
    }
}
```

Output:

```
Enter the number of elements: 6
Enter 6 integers:
23
12
4
89
56
1
Sorted array in ascending order:
1 4 12 23 56 89
```

1.f) Reverse Array Elements

Develop a Java program that takes input for an array of integers and prints the elements in reverse order using a loop.

Aim:

To write a Java program that accepts an array of integers from the user and prints the elements in **reverse order** using a loop.

Algorithm:

1. Start the program.
2. Ask the user to input the number of elements.
3. Read the elements into an array.
4. Use a for loop to print the elements in reverse order (from last index to 0).
5. End the program.

Program:

```
import java.util.Scanner;

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

```
        // Step 1: Input array size
        System.out.print("Enter the number of elements: ");
        int n = scanner.nextInt();

        int[] array = new int[n];

        // Step 2: Input array elements
        System.out.println("Enter " + n + " integers:");
        for (int i = 0; i < n; i++) {
            array[i] = scanner.nextInt();
        }

        // Step 3: Print array in reverse order
        System.out.println("Array elements in reverse order:");
        for (int i = n - 1; i >= 0; i--) {
            System.out.print(array[i] + " ");
        }

        scanner.close();
    }
}
```

Output:

```
Enter the number of elements: 5
Enter 5 integers:
10
20
30
40
50
Array elements in reverse order:
50 40 30 20 10
```

1.g) Multiplication Table Generator Using 2D Arrays

Create a Java program that uses a 2D array to store and display multiplication tables from 1 to 10 using nested loops.

Aim:

To write a Java program that generates multiplication tables from **1 to 10** using a **2D array** and displays the result using nested loops.

Algorithm:

1. Start the program.
2. Declare a 2D array table[10][10] to store multiplication results from 1 to 10.
3. Use nested loops:
    - Outer loop for rows (1 to 10)
    - Inner loop for columns (1 to 10)
    - Compute table[i][j] = (i+1) * (j+1)
4. Use another nested loop to display the table in a formatted way.
5. End the program.

Program:

```
public class MultiplicationTable2D {
    public static void main(String[] args) {
        int[][] table = new int[10][10];
```

```
        // Step 1: Generate multiplication table
        for (int i = 0; i < 10; i++) {
            for (int j = 0; j < 10; j++) {
                table[i][j] = (i + 1) * (j + 1);
            }
        }

        // Step 2: Display the table
        System.out.println("Multiplication Table (1 to 10):");
        for (int i = 0; i < 10; i++) {
            for (int j = 0; j < 10; j++) {
                System.out.printf("%4d", table[i][j]);
            }
            System.out.println();
        }
    }
}
```

Output:

Multiplication Table (1 to 10):

```
   1   2   3   4   5   6   7   8   9  10
   2   4   6   8  10  12  14  16  18  20
   3   6   9  12  15  18  21  24  27  30
   4   8  12  16  20  24  28  32  36  40
   5  10  15  20  25  30  35  40  45  50
   6  12  18  24  30  36  42  48  54  60
   7  14  21  28  35  42  49  56  63  70
   8  16  24  32  40  48  56  64  72  80
   9  18  27  36  45  54  63  72  81  90
  10  20  30  40  50  60  70  80  90 100
```

1.h) Frequency of Elements in an Array

Write a Java program to find and print the frequency of each element in an array using a combination of arrays and nested loops.

Aim:

To write a Java program that calculates and displays the frequency of each element in an array using nested loops and a helper array.

Algorithm:

1. Start the program.
2. Ask the user to input the size of the array.
3. Read the array elements from the user.
4. Create a visited[] array of the same size, initialized to false or -1.
5. Use a nested loop:
    - Outer loop iterates over each element.
    - If the element is not already counted:
        - Set frequency = 1.
        - Inner loop checks for duplicates and increments the frequency.
        - Mark all duplicates in visited[].
6. Print the frequency of each unique element.
7. End the program.

Program:

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

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

      // Step 1: Input array size
      System.out.print("Enter the number of elements: ");
      int n = scanner.nextInt();

      int[] array = new int[n];
      int[] frequency = new int[n]; // Frequency tracker array
      int visited = -1;

      // Step 2: Input array elements
      System.out.println("Enter " + n + " integers:");
      for (int i = 0; i < n; i++) {
         array[i] = scanner.nextInt();
      }

      // Step 3: Calculate frequency using nested loops
      for (int i = 0; i < n; i++) {
         if (frequency[i] == visited)
            continue;

         int count = 1;
         for (int j = i + 1; j < n; j++) {
            if (array[i] == array[j]) {
               count++;
               frequency[j] = visited; // Mark as visited
            }
         }
         frequency[i] = count;
      }

      // Step 4: Display frequencies
      System.out.println("Frequency of each element:");
      for (int i = 0; i < n; i++) {
         if (frequency[i] != visited) {
            System.out.println(array[i] + " occurs " + frequency[i] + " times");
         }
      }

      scanner.close();
   }
}
```

Output:

```
Enter the number of elements: 7
Enter 7 integers:
4 5 6 4 5 7 4
Frequency of each element:
4 occurs 3 times
5 occurs 2 times
6 occurs 1 times
7 occurs 1 times
```

1.i) Menu-Driven Array Operations Program

Design a menu-driven Java application that allows users to perform operations like insert, delete, update, and display elements in an array using switch statements.

Aim:

To design a menu-driven Java program that allows the user to perform array operations — **insert**, **delete**, **update**, and **display** — using switch statements.

Algorithm:

1. Start the program.
2. Initialize an array with a fixed size and track the current number of elements.
3. Display a menu with options:
    - Insert an element
    - Delete an element by value
    - Update an element at a specific index
    - Display all elements
    - Exit
4. Take the user's choice.
5. Use a switch statement to handle each choice:
    - **Insert**: Add element at the end if space available.
    - **Delete**: Find element by value and remove it (shift elements left).
    - **Update**: Replace element at given index with a new value.
    - **Display**: Print all current elements.
    - **Exit**: Terminate the program.
6. Repeat until the user chooses to exit.
7. End the program.

Program:

```
import java.util.Scanner;

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

      int[] array = new int[100]; // max size 100
      int size = 0; // current number of elements
      boolean running = true;

      while (running) {
         System.out.println("\n--- Menu ---");
         System.out.println("1. Insert");
         System.out.println("2. Delete");
         System.out.println("3. Update");
         System.out.println("4. Display");
         System.out.println("5. Exit");
         System.out.print("Enter your choice: ");

         int choice = scanner.nextInt();

         switch (choice) {
            case 1:
               if (size == array.length) {
                  System.out.println("Array is full! Cannot insert new element.");
```

```
    } else {
        System.out.print("Enter element to insert: ");
        int element = scanner.nextInt();
        array[size] = element;
        size++;
        System.out.println("Element inserted.");
    }
    break;

case 2:
    if (size == 0) {
        System.out.println("Array is empty! Nothing to delete.");
    } else {
        System.out.print("Enter element to delete: ");
        int element = scanner.nextInt();
        int index = -1;
        for (int i = 0; i < size; i++) {
            if (array[i] == element) {
                index = i;
                break;
            }
        }
        if (index == -1) {
            System.out.println("Element not found in array.");
        } else {
            // Shift elements left to delete
            for (int i = index; i < size - 1; i++) {
                array[i] = array[i + 1];
            }
            size--;
            System.out.println("Element deleted.");
        }
    }
    break;

case 3:
    if (size == 0) {
        System.out.println("Array is empty! Nothing to update.");
    } else {
        System.out.print("Enter index to update (0 to " + (size - 1) + "): ");
        int index = scanner.nextInt();
        if (index < 0 || index >= size) {
            System.out.println("Invalid index!");
        } else {
            System.out.print("Enter new value: ");
            int newValue = scanner.nextInt();
            array[index] = newValue;
            System.out.println("Element updated.");
        }
    }
    break;

case 4:
    if (size == 0) {
        System.out.println("Array is empty.");
```

```
                } else {
                    System.out.print("Array elements: ");
                    for (int i = 0; i < size; i++) {
                        System.out.print(array[i] + " ");
                    }
                    System.out.println();
                }
                break;

            case 5:
                running = false;
                System.out.println("Exiting program. Goodbye!");
                break;

            default:
                System.out.println("Invalid choice! Please try again.");
        }
    }

    scanner.close();
  }
}
```

Output:

```
--- Menu ---
1. Insert
2. Delete
3. Update
4. Display
5. Exit
Enter your choice: 1
Enter element to insert: 10
Element inserted.

--- Menu ---
1. Insert
2. Delete
3. Update
4. Display
5. Exit
Enter your choice: 1
Enter element to insert: 20
Element inserted.

--- Menu ---
1. Insert
2. Delete
3. Update
4. Display
5. Exit
Enter your choice: 4
Array elements: 10 20

--- Menu ---
1. Insert
2. Delete
```

```
3. Update
4. Display
5. Exit
Enter your choice: 3
Enter index to update (0 to 1): 1
Enter new value: 25
Element updated.

--- Menu ---
1. Insert
2. Delete
3. Update
4. Display
5. Exit
Enter your choice: 4
Array elements: 10 25

--- Menu ---
1. Insert
2. Delete
3. Update
4. Display
5. Exit
Enter your choice: 2
Enter element to delete: 10
Element deleted.

--- Menu ---
1. Insert
2. Delete
3. Update
4. Display
5. Exit
Enter your choice: 4
Array elements: 25

--- Menu ---
1. Insert
2. Delete
3. Update
4. Display
5. Exit
Enter your choice: 5
Exiting program. Goodbye!
```

1.j) Prime Number Finder in Array

Develop a Java program that checks each element of an integer array and prints all the prime numbers using loops and conditional logic.

Aim:

To develop a Java program that checks each element of an integer array and prints all the prime numbers using loops and conditional statements.

Algorithm:

1. Start the program.
2. Take input for the number of elements.
3. Read elements into the array.
4. For each element in the array:
    - Check if it is prime by testing divisibility from 2 to √element.
    - If it's prime, print the element.
5. End the program.

Program:

```
import java.util.Scanner;

public class PrimeNumberFinder {

   // Method to check if a number is prime
   public static boolean isPrime(int num) {
      if (num <= 1) return false; // 0 and 1 are not prime

      for (int i = 2; i <= Math.sqrt(num); i++) {
         if (num % i == 0) return false; // divisible, not prime
      }
      return true; // prime number
   }

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

      // Step 1: Input number of elements
      System.out.print("Enter the number of elements: ");
      int n = scanner.nextInt();

      int[] array = new int[n];

      // Step 2: Input elements
      System.out.println("Enter " + n + " integers:");
      for (int i = 0; i < n; i++) {
         array[i] = scanner.nextInt();
      }

      // Step 3: Find and print prime numbers
      System.out.println("Prime numbers in the array are:");
      boolean foundPrime = false;
      for (int num : array) {
         if (isPrime(num)) {
            System.out.print(num + " ");
            foundPrime = true;
         }
      }
      if (!foundPrime) {
```

```
            System.out.println("No prime numbers found.");
        }

        scanner.close();
    }
}
```

Output:

```
Enter the number of elements: 7
Enter 7 integers:
4 5 6 7 8 9 11
Prime numbers in the array are:
5 7 11
```