JAVA LAB 2

Q.1 Print given number in words ?

```
-> import java.util.Scanner;
   public class Word {
  // Array for number words
  private static final String[] units = {
       "", "One", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", "Nine"
  };
  private static final String[] teens = {
       "Ten", "Eleven", "Twelve", "Thirteen", "Fourteen", "Fifteen",
       "Sixteen", "Seventeen", "Eighteen", "Nineteen"
  };
  private static final String[] tens = {
       "", "", "Twenty", "Thirty", "Forty", "Fifty", "Sixty", "Seventy", "Eighty", "Ninety"
  };
  private static final String[] thousands = {
       "", "Thousand", "Million", "Billion"
  };  // Main method
  public static void main(String[] args) {
     Scanner scanner = new Scanner(System.in);
     System.out.println("Enter a number: ");
     int number = scanner.nextInt();
     scanner.close();
    if (number == 0) {
    System.out.println("Zero");
     } else {
        System.out.println(numberToWords(number));
```

```
    } }
// Method to convert number to words
private static String numberToWords(int number) {
    if (number == 0) {
        return "Zero";
    }
    int thousandCounter = 0;
    String words = "";
    while (number > 0) {
        if (number % 1000 != 0) {
            words = convertBelowThousand(number % 1000) + thousands[thousandCounter] + " " + words;
        }
        number /= 1000;
        thousandCounter++;
    }
    return words.trim();
}
// Method to convert numbers below one thousand
private static String convertBelowThousand(int number) {
    String str = "";
    if (number % 100 < 20) {
        str = number % 100 < 10 ? units[number % 10] : teens[number % 10];
        number /= 100;
    } else {
        str = units[number % 10];
        number /= 10;
        str = tens[number % 10] + " " + str;
        number /= 10;
```

```
    }
    if (number == 0) {
       return str;
    }
    return units[number] + " Hundred " + str;
  }}
```

Output :- Enter a number: 12

Twelve

Q.2. Program to check the given number is palindrome or not .

```
-> import java.util.Scanner;
public class PrimeNumbers {
  // Method to check if a number is prime
  public static boolean isPrime(int num) {
    if (num < 2) return false;
    for (int i = 2; i <= num / 2; i++) {
       if (num % i == 0) return false;
    }
    return true;   }  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    // Input N
    System.out.print("Enter the number of prime numbers to display: ");
    int N = scanner.nextInt();
    int count = 0, num = 2, sum = 0;
    System.out.println("The first " + N + " prime numbers are:");
    // Find and display the first N prime numbers
    while (count < N) {
       if (isPrime(num)) {
```

```
            System.out.print(num + " ");
            sum += num;
            count++;
        }
        num++;
    }
    // Calculate and display the sum and average
    double average = (double) sum / N;
    System.out.println("\nSum: " + sum);
    System.out.println("Average: " + average);
    scanner.close();
}}
```

Output :- Enter the number of prime numbers to display: 12

The first 12 prime numbers are: 2 3 5 7 11 13 17 19 23 29 31 37 ,

Sum: 197 , Average: 16.416666666666668

Q.3. Print Floyd's Triangle ?

->
```
import java.util.Scanner;

public class FloydsTriangle {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Input the number of rows for Floyd's Triangle

System.out.print("Enter the number of rows for Floyd's Triangle: ");

int rows = scanner.nextInt();

int number = 1; // Start from 1

System.out.println("Floyd's Triangle:");

// Print Floyd's Triangle

for (int i = 1; i <= rows; i++) {

    for (int j = 1; j <= i; j++) {

        System.out.print(number + " ");

        number++;

    }

    System.out.println();
```

```
    }
    scanner.close();
  }}
```

// Output:

Enter the number of rows for Floyd's Triangle: 5

1

2 3

4 5 6

7 8 9 10

11 12 13 14 15

Q.4. Program to find second largest number in an array

```
-> import java.util.Scanner;
public class SecondLargestNumber {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      // Input the size of the array
      System.out.print("Enter the size of the array: ");
      int size = scanner.nextInt();
      int[] array = new int[size];
      // Input the elements of the array
      System.out.println("Enter the elements of the array:");
      for (int i = 0; i < size; i++) {
         array[i] = scanner.nextInt();
      }
      // Find the largest and second largest numbers
      int largest = Integer.MIN_VALUE;
      int secondLargest = Integer.MIN_VALUE;
```

```
        for (int i = 0; i < size; i++) {
            if (array[i] > largest) {
                secondLargest = largest;
                largest = array[i];
            } else if (array[i] > secondLargest && array[i] != largest) {
                secondLargest = array[i];
            } }
        // Output the second largest number
        if (secondLargest == Integer.MIN_VALUE) {
            System.out.println("There is no second largest element in the array.");
        } else {
            System.out.println("The second largest number in the array is: " + secondLargest);
        }
        scanner.close();
    }}
```

Output :- Enter the first number: 15

Enter the second number: 12

HCF of 15 and 12 is: 3

LCM of 15 and 12 is: 60

Q.5. Program to count number of words in given string .

->

```
import java.util.Scanner;
public class NumberOfWords {
public static int countWords(String str) {
 if (str == null || str.isEmpty()) {
    return 0;
```

```
        }
        // Split the string by whitespace
        String[] words = str.split("\\s+");
        // Return the number of words
        return words.length;
    }
public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        // Input the string
        System.out.print("Enter a string: ");
        String input = scanner.nextLine();
        // Count the number of words
        int wordCount = countWords(input);
        // Display the result
        System.out.println("Number of words in the string: " + wordCount);
        scanner.close();
    }}
```

Output :- Enter a string: Ram

Number of words in the string: 1

Q.6. Program to find difference of minimum and maximum numbers of array in java ?

->

```
public class MinMaxDifference {
    public static void main(String[] args) {
        // Example array
        int[] numbers = {3, 5, 7, 2, 8, -1, 4, 10, 12};
        // Find the minimum and maximum numbers in the array
```

```
    int min = findMin(numbers);
    int max = findMax(numbers);
    // Calculate the difference
    int difference = max - min;
    // Print the result
    System.out.println("Minimum number: " + min);
    System.out.println("Maximum number: " + max);
    System.out.println("Difference: " + difference);
}
// Method to find the minimum number in an array
public static int findMin(int[] array) {
    int min = array[0];
    for (int num : array) {
        if (num < min) {
            min = num;
        }
    }
    return min;
}
// Method to find the maximum number in an array
public static int findMax(int[] array) {
    int max = array[0];
    for (int num : array) {
        if (num > max) {
            max = num;
        } }
    return max;
}}
```

Output :- Minimum number: -1

Maximum number: 12

Difference: 13

Q.7. Implement Stack Operation

->
```
public class Stack {
   private int[] stackArray;
   private int top;
   private int maxSize;

   // Constructor to initialize the stack
   public Stack(int size) {
      maxSize = size;
      stackArray = new int[maxSize];
      top = -1;
   }
   // Method to add an element to the stack
   public void push(int value) {
      if (isFull()) {
         System.out.println("Stack is full. Cannot push " + value);
      } else {
         stackArray[++top] = value;
      }
   }
   // Method to remove and return the top element from the stack
   public int pop() {
      if (isEmpty()) {
         System.out.println("Stack is empty. Cannot pop");
         return -1;
      } else {
         return stackArray[top--];
      }
   }
   // Method to peek at the top element without removing it
   public int peek() {
      if (isEmpty()) {
         System.out.println("Stack is empty. Cannot peek");
         return -1;
      } else {
         return stackArray[top];
```

```
        }
    }
    // Method to check if the stack is empty
    public boolean isEmpty() {
        return (top == -1);
    }
    // Method to check if the stack is full
    public boolean isFull() {
        return (top == maxSize - 1);
    }
    // Main method to demonstrate stack operations
    public static void main(String[] args) {
        Stack stack = new Stack(5);
        // Push elements to the stack
        stack.push(10);
        stack.push(20);
        stack.push(30);
        stack.push(40);
        stack.push(50);
        stack.push(60); // This will show that the stack is full

        // Peek at the top element
        System.out.println("Top element is: " + stack.peek());

        // Pop elements from the stack
        System.out.println("Popped element: " + stack.pop());
        System.out.println("Popped element: " + stack.pop());
        System.out.println("Popped element: " + stack.pop());
        // Peek at the top element after pops
        System.out.println("Top element is: " + stack.peek());
    }
}
```

Output :- Stack is full. Cannot push 60

Top element is: 50

Popped element: 50

Popped element: 40

Popped element: 30

Top element is: 20

Q.8. Implement Queue Operation

```
->  public class Queue {
    private int[] queueArray;
    private int front;
    private int rear;
    private int maxSize;
    private int currentSize;
    // Constructor to initialize the queue
    public Queue(int size) {
        maxSize = size;
        queueArray = new int[maxSize];
        front = 0;
        rear = -1;
        currentSize = 0;
    }
    // Method to add an element to the queue
    public void enqueue(int value) {
        if (isFull()) {
            System.out.println("Queue is full. Cannot enqueue " + value);
        } else {
            rear = (rear + 1) % maxSize;
            queueArray[rear] = value;
            currentSize++;
        }
    }
    // Method to remove and return the front element from the queue
    public int dequeue() {
        if (isEmpty()) {
            System.out.println("Queue is empty. Cannot dequeue");
            return -1;
        } else {
            int value = queueArray[front];
            front = (front + 1) % maxSize;
            currentSize--;
            return value;
        }
    }
    // Method to peek at the front element without removing it
    public int peek() {
        if (isEmpty()) {
            System.out.println("Queue is empty. Cannot peek");
```

```
            return -1;
        } else {
            return queueArray[front];
        }
    }
    // Method to check if the queue is empty
    public boolean isEmpty() {
        return (currentSize == 0);
    }
    // Method to check if the queue is full
    public boolean isFull() {
        return (currentSize == maxSize);
    }
    // Main method to demonstrate queue operations
    public static void main(String[] args) {
        Queue queue = new Queue(5);
        // Enqueue elements to the queue
        queue.enqueue(10);
        queue.enqueue(20);
        queue.enqueue(30);
        queue.enqueue(40);
        queue.enqueue(50);
        queue.enqueue(60); // This will show that the queue is full
        // Peek at the front element
        System.out.println("Front element is: " + queue.peek());
        // Dequeue elements from the queue
        System.out.println("Dequeued element: " + queue.dequeue());
        System.out.println("Dequeued element: " + queue.dequeue());
        System.out.println("Dequeued element: " + queue.dequeue());
        // Peek at the front element after dequeues
        System.out.println("Front element is: " + queue.peek());
    }
}
```

Output :- Queue is full. Cannot enqueue 60

Front element is: 10

Dequeued element: 10

Dequeued element: 20

Dequeued element: 30

Front element is: 40

Q.9 Bubble Sort ?

->
```
public class BubbleSort {
  // Method to perform Bubble Sort on an array
  public static void bubbleSort(int[] array) {
    int n = array.length;
    boolean swapped;

    // Loop through the array
    for (int i = 0; i < n - 1; i++) {
      swapped = false;

      // Inner loop to compare adjacent elements
      for (int j = 0; j < n - 1 - i; j++) {
        if (array[j] > array[j + 1]) {
          // Swap the elements if they are in the wrong order
          int temp = array[j];
          array[j] = array[j + 1];
          array[j + 1] = temp;
          swapped = true;
        }
      }
      // If no elements were swapped, the array is already sorted
      if (!swapped) {
        break;
      }
    }
  }
  // Main method to test the Bubble Sort
  public static void main(String[] args) {
    int[] numbers = {64, 34, 25, 12, 22, 11, 90};
    System.out.println("Unsorted array:");
    printArray(numbers);
    bubbleSort(numbers);
    System.out.println("Sorted array:");
    printArray(numbers);
  }
  // Method to print the elements of an array
  public static void printArray(int[] array) {
    for (int num : array) {
```

```
        System.out.print(num + " ");
      }
      System.out.println();
   }
}
```

Output :- Unsorted array:
64 34 25 12 22 11 90
Sorted array:
11 12 22 25 34 64 90

Q.10. Selection Sort ?

->
```
public class SelectionSort {
   // Method to perform Selection Sort on an array
   public static void selectionSort(int[] array) {
      int n = array.length;
      // Loop through the array
      for (int i = 0; i < n - 1; i++) {
         // Assume the minimum is the first element
         int minIndex = i;
         // Find the index of the minimum element in the remaining unsorted array
         for (int j = i + 1; j < n; j++) {
            if (array[j] < array[minIndex]) {
               minIndex = j;
            }
         }
         // Swap the found minimum element with the first element of the unsorted part
         int temp = array[minIndex];
         array[minIndex] = array[i];
         array[i] = temp;
      }}
   // Main method to test the Selection Sort
   public static void main(String[] args) {
      int[] numbers = {64, 25, 12, 22, 11};
      System.out.println("Unsorted array:");
      printArray(numbers);
      selectionSort(numbers);
      System.out.println("Sorted array:");
      printArray(numbers);
   }
   // Method to print the elements of an array
   public static void printArray(int[] array) {
      for (int num : array) {
```

```
        System.out.print(num + " ");
    System.out.println();   } }
```

Output :-

```
Unsorted array:
64 25 12 22 11
Sorted array:
11 12 22 25 64
```

Q.11. Insertion Sort

```
-> public class InsertionSort {
   // Method to perform Insertion Sort on an array
   public static void insertionSort(int[] array) {
      int n = array.length;
      // Loop through the array starting from the second element
      for (int i = 1; i < n; i++) {
         int key = array[i];
         int j = i - 1;
         // Move elements of array[0..i-1] that are greater than the key
         // to one position ahead of their current position
         while (j >= 0 && array[j] > key) {
            array[j + 1] = array[j];
            j = j - 1;
         }
         array[j + 1] = key;
      }
   }
   // Main method to test the Insertion Sort
   public static void main(String[] args) {
      int[] numbers = {12, 11, 13, 5, 6};
      System.out.println("Unsorted array:");
      printArray(numbers);
      insertionSort(numbers);
      System.out.println("Sorted array:");
      printArray(numbers);
   }
   // Method to print the elements of an array
   public static void printArray(int[] array) {
      for (int num : array) {
         System.out.print(num + " ");
      }
```

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

Output :-

```
Unsorted array:
12 11 13 5 6
Sorted array:
5 6 11 12 13
```

Q.12 Quick Sort

```
-> public class QuickSort {
    // Method to perform Quick Sort on an array
    public static void quickSort(int[] array, int low, int high) {
        if (low < high) {
            // Find the partition index
            int partitionIndex = partition(array, low, high);

            // Recursively sort the elements before and after partition
            quickSort(array, low, partitionIndex - 1);
            quickSort(array, partitionIndex + 1, high);
        }
    }
    // Method to partition the array and return the partition index
    public static int partition(int[] array, int low, int high) {
        // Choose the rightmost element as pivot
        int pivot = array[high];
        int i = (low - 1); // Index of smaller element
        for (int j = low; j < high; j++) {
            // If the current element is smaller than or equal to the pivot
            if (array[j] <= pivot) {
                i++;
                // Swap array[i] and array[j]
                int temp = array[i];
                array[i] = array[j];
                array[j] = temp;
            }
        }
        // Swap array[i + 1] and array[high] (or pivot)
        int temp = array[i + 1];
        array[i + 1] = array[high];
        array[high] = temp;
        return i + 1;
    }
```

```
  // Main method to test the Quick Sort
  public static void main(String[] args) {
    int[] numbers = {10, 7, 8, 9, 1, 5};
    System.out.println("Unsorted array:");
    printArray(numbers);
    quickSort(numbers, 0, numbers.length - 1);
    System.out.println("Sorted array:");
    printArray(numbers);
  }
  // Method to print the elements of an array
  public static void printArray(int[] array) {
    for (int num : array) {
      System.out.print(num + " ");
    }
    System.out.println();
  }}
```

Output :-

```
Unsorted array:
10 7 8 9 1 5
Sorted array:
1 5 7 8 9 10
```

Q.13 Program to Calculate HCF and LCM

```
-> import java.util.Scanner;
public class HCFandLCM {
  // Method to calculate HCF of two numbers
  public static int calculateHCF(int num1, int num2) {
    while (num2 != 0) {
      int temp = num2;
      num2 = num1 % num2;
      num1 = temp;
    }
    return num1;
  }
  // Method to calculate LCM of two numbers using HCF
  public static int calculateLCM(int num1, int num2) {
    int hcf = calculateHCF(num1, num2);
    int lcm = (num1 * num2) / hcf;
    return lcm;
  }
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
```

```
        // Input the two numbers
        System.out.print("Enter the first number: ");
        int num1 = scanner.nextInt();
        System.out.print("Enter the second number: ");
        int num2 = scanner.nextInt();
        // Calculate and display HCF
        int hcf = calculateHCF(num1, num2);
        System.out.println("HCF of " + num1 + " and " + num2 + " is: " + hcf);
        // Calculate and display LCM
        int lcm = calculateLCM(num1, num2);
        System.out.println("LCM of " + num1 + " and " + num2 + " is: " + lcm);
        scanner.close();
    }
}
```

Output :- Enter the first number: 12
Enter the second number: 13
HCF of 12 and 13 is: 1
LCM of 12 and 13 is: 156

Q.14. Program to find largest and second largest in an array

->

```
import java.util.Scanner;
public class LargestAndSecondLargest {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        // Input the size of the array
        System.out.print("Enter the size of the array: ");
        int size = scanner.nextInt();
        if (size < 2) {
            System.out.println("Array should have at least two elements.");
            return;
        }
        int[] array = new int[size];
        // Input the elements of the array
        System.out.println("Enter the elements of the array:");
        for (int i = 0; i < size; i++) {
            array[i] = scanner.nextInt();
        }
        // Find the largest and second largest numbers
        int largest = Integer.MIN_VALUE;
        int secondLargest = Integer.MIN_VALUE;
        for (int i = 0; i < size; i++) {
            if (array[i] > largest) {
```

```
        secondLargest = largest;
        largest = array[i];
    } else if (array[i] > secondLargest && array[i] != largest) {
        secondLargest = array[i];
    }
}
// Output the largest and second largest numbers
if (secondLargest == Integer.MIN_VALUE) {
    System.out.println("There is no second largest element in the array.");
} else {
    System.out.println("The largest number in the array is: " + largest);
    System.out.println("The second largest number in the array is: " + secondLargest);
}
scanner.close();  } }
```

Output :-

```
Enter the size of the array: 4
Enter the elements of the array:
12
18
20
22
The largest number in the array is: 22
The second largest number in the array is: 20
```