

ASSIGNMENT-7

Subject – Computer Science Workshop-1(CSE 2141)

NAME – DEEPTESH ROUT

Registration Number – 2341018166

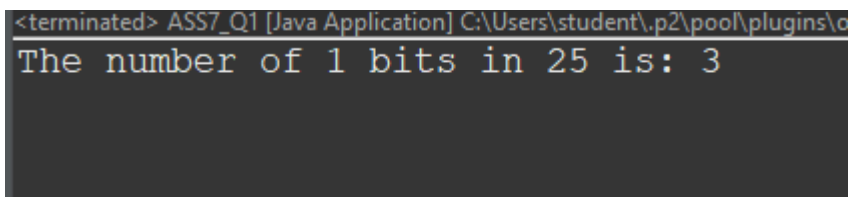
Section – 23412G1 Branch – B. Tech (CSE)

/*Q1 Write a Java program to count the number of bits that are set to 1 in an integer.

JAVA CODE

```
public class ASS7_Q1 {  
  
    // Function to count the number of 1 bits in an integer  
    public static int countSetBits(int number) {  
        int count = 0;  
        while (number != 0) {  
            count += number & 1; // Increment count if the last bit is 1  
            number >>= 1;      // Unsigned right shift  
        }  
        return count;  
    }  
  
    public static void main(String[] args) {  
        // Example usage  
        int number = 29; // Binary representation: 11101  
        int result = countSetBits(number);  
  
        System.out.println("The number of 1 bits in " + number + " is: " + result);  
    }  
}
```

OUTPUT



The screenshot shows a Java application window titled "<terminated> ASS7_Q1 [Java Application] C:\Users\student\.p2\pool\plugins\c". The output text in the window is "The number of 1 bits in 25 is: 3".

Q2 The parity of a binary word is 1 if the number of 1s in the word is odd; otherwise, it is 0. Write a Java program to count the parity of an integer number.

JAVA CODE :-

```
import java.util.Scanner;

public class ASS7_Q2 {
    // Method to calculate the parity of a number
    public static int calculateParity(int number) {
        int count = 0;

        // Count the number of 1s in the binary representation
        while (number != 0) {
            count += number & 1; // Add 1 if the least significant bit is 1
            number >>= 1;       // Right shift the number by 1 bit
        }

        // Return 1 if the count is odd, 0 otherwise
        return count % 2;
    }

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

        // Input: Get the integer from the user
        System.out.print("Enter an integer: ");
        int number = scanner.nextInt();

        // Calculate the parity
        int parity = calculateParity(number);

        // Output the result
        System.out.println("The parity of the number is: " + parity);

        scanner.close();
    }
}
```

OUTPUT:-

```
Enter an integer: 45
The parity of the number is: Even
```

Q3. Write a program to swap the i-th bit with j-th bit of a number.

JAVA CODE :-

```
public class ASS7_Q3 {

    public static int swapBits(int number, int i, int j) {
        // Extract the i-th and j-th bits
        int bit1 = (number >> i) & 1;
        int bit2 = (number >> j) & 1;

        // If the bits are the same, no need to swap
        if (bit1 == bit2) {
            return number;
        }

        // Create a bitmask with the i-th and j-th bits set
        int bitMask = (1 << i) | (1 << j);

        // Toggle the i-th and j-th bits using XOR
        return number ^ bitMask;
    }

    public static void main(String[] args) {
        int number = 73; // Example number (binary: 1001001)
        int i = 1;       // Position of first bit (0-based indexing)
        int j = 6;       // Position of second bit (0-based indexing)

        System.out.println("Original number: " + number + " (Binary: " +
            Integer.toBinaryString(number) + ")");
        int result = swapBits(number, i, j);
        System.out.println("After swapping: " + result + " (Binary: " +
            Integer.toBinaryString(result) + ")");
    }
}
```

OUTPUT:-

```
Original number: 14 (Binary: 1110)
After swapping: 28 (Binary: 11100)
```

Q4. Write a program that takes a 64-bit word and returns the 64-bit word consisting of the bits of the input word in reverse order. For example, if the input is alternating 1s and 0s, i.e., (1010...10), the output should be alternating 0s and 1s, i.e.,(0101...01).

JAVA CODE

```
public class ASS7_Q4 {

    public static long reverseBits(long number) {
        long reversed = 0; // Initialize the reversed number
        for (int i = 0; i < 64; i++) {
            // Extract the least significant bit of the number
            long bit = number & 1;

            // Shift the bit to its reversed position and add it to reversed
            reversed = (reversed << 1) | bit;

            // Shift the input number to the right to process the next bit
            number >>= 1;
        }
        return reversed;
    }
}
```

```
public static void main(String[] args) {
    long input = 0xAAAAAAAAAAAAAAAAAL; // Example: alternating 1s and 0s
    System.out.println("Input (Binary): " + Long.toBinaryString(input));
    long output = reverseBits(input);
    System.out.println("Output (Binary): " + Long.toBinaryString(output));
}
```

OUTPUT :-

[illegible]

Q5. Write a java program to compute $x \times y$ without arithmetic operators.

JAVA CODE

```
public class ASS7_Q5 {

    public static int multiply(int x, int y) {
        int result = 0; // Initialize result to 0

        // Iterate through all bits of y
        while (y != 0) {
            // If the least significant bit of y is set, add x to the result
            if ((y & 1) != 0) {
                result = add(result, x);
            }

            // Shift x to the left (equivalent to multiplying by 2)
            x <<= 1;

            // Shift y to the right (equivalent to dividing by 2)
            y >>= 1;
        }

        return result;
    }

    // Helper method to add two integers without using the '+' operator
    private static int add(int a, int b) {
        while (b != 0) {
            // Calculate carry
            int carry = a & b;

            // Perform addition without carry
            a = a ^ b;

            // Shift carry left by 1 to add it in the next iteration
            b = carry << 1;
        }
        return a;
    }

    public static void main(String[] args) {
        int x = 7; // Example input
        int y = 3; // Example input

        System.out.println("x: " + x + ", y: " + y);
        System.out.println("x * y: " + multiply(x, y));
    }
}
```

OUTPUT :-

```
x: 7, y: 3  
x * y: 21
```

Q6. Write a java program to compute x/y without arithmetic operators.

JAVA CODE :-

```
public class ASS7_Q6 {  
  
    public static int divide(int dividend, int divisor) {  
        // Handle edge cases for division by 0  
        if (divisor == 0) {  
            throw new ArithmeticException("Division by zero is undefined.");  
        }  
  
        // Handle overflow case when dividing Integer.MIN_VALUE by -1  
        if (dividend == Integer.MIN_VALUE && divisor == -1) {  
            return Integer.MAX_VALUE;  
        }  
  
        // Determine the sign of the result  
        boolean negative = (dividend < 0) ^ (divisor < 0);  
  
        // Work with positive values for simplicity  
        long absDividend = Math.abs((long) dividend);  
        long absDivisor = Math.abs((long) divisor);  
  
        int result = 0;  
  
        // Perform division using bitwise operations  
        while (absDividend >= absDivisor) {  
            long tempDivisor = absDivisor;  
            int multiple = 1;  
  
            // Shift tempDivisor left until it's greater than absDividend  
            while (absDividend >= (tempDivisor << 1)) {  
                tempDivisor <<= 1;  
                multiple <<= 1;  
            }  
  
            // Subtract tempDivisor from absDividend and add multiple to result  
            absDividend -= tempDivisor;
```



```

        result += multiple;
    }

    // Apply the sign to the result
    return negative ? -result : result;
}

public static void main(String[] args) {
    int dividend = 43; // Example dividend
    int divisor = 5; // Example divisor

    System.out.println("Dividend: " + dividend + ", Divisor: " + divisor);
    System.out.println("Quotient: " + divide(dividend, divisor));
}
}

```

OUTPUT :-

```

Dividend: 43, Divisor: 5
Quotient: 8

```

Q7. Write a program to find x^y .

JAVA CODE

```
public class ASS7_Q7 {

    public static long power(int x, int y) {
        if (y < 0) {
            throw new IllegalArgumentException("Negative powers are not supported.");
        }

        long result = 1;
        long base = x;

        // Perform exponentiation using bitwise operations
        while (y > 0) {
            // If the current bit of y is 1, multiply result by base
            if ((y & 1) == 1) {
                result *= base;
            }

            // Square the base and shift y to the right
            base *= base;
            y >>= 1;
        }

        return result;
    }

    public static void main(String[] args) {
        int x = 3; // Base
        int y = 4; // Exponent

        System.out.println(x + " raised to the power " + y + " is: " + power(x, y));
    }
}
```

OUTPUT :-

```
3 raised to the power 4 is: 81
```

Q8. Write a program to find the reverse of a number. For example, if the input is 123 output is 321, and if the input is -245 output is -542

JAVA CODE :-

```
public class ASS7_Q8 {

    public static int reverse(int number) {
        int reversed = 0;

        while (number != 0) {
            // Extract the last digit
            int digit = number % 10;

            // Check for overflow/underflow before adding the digit
            if (reversed > Integer.MAX_VALUE / 10 || reversed < Integer.MIN_VALUE / 10) {
                throw new ArithmeticException("Overflow occurred during reversal.");
            }

            // Add the digit to the reversed number
            reversed = reversed * 10 + digit;

            // Remove the last digit from the number
            number /= 10;
        }

        return reversed;
    }

    public static void main(String[] args) {
        int input = -245; // Example input
        System.out.println("Original number: " + input);
        System.out.println("Reversed number: " + reverse(input));
    }
}
```

OUTPUT :-

```
Original number: -245  
Reversed number: -542
```

Q9. Write a program to check whether a number is palindrome or not.

JAVA CODE :-

```
public class ASS7_Q9 {  
  
    public static boolean isPalindrome(int number) {  
        // Negative numbers are not palindromes  
        if (number < 0) {  
            return false;  
        }  
  
        int original = number; // Store the original number  
        int reversed = 0;  
  
        // Reverse the number  
        while (number != 0) {  
            int digit = number % 10;  
  
            // Check for overflow  
            if (reversed > Integer.MAX_VALUE / 10) {
```

```

        return false; // Overflow indicates it's not a valid palindrome
    }

    reversed = reversed * 10 + digit;
    number /= 10;
}

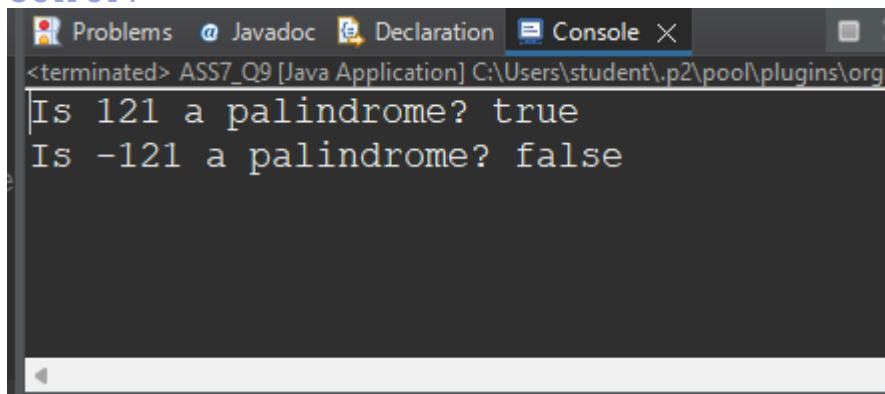
// Check if the reversed number equals the original
return original == reversed;
}

public static void main(String[] args) {
    int input = 121; // Example input
    System.out.println("Is " + input + " a palindrome? " + isPalindrome(input));

    int input2 = -121; // Another example input
    System.out.println("Is " + input2 + " a palindrome? " + isPalindrome(input2));
}
}

```

OUTPUT :-



The screenshot shows a Java IDE window with a console tab. The console output is as follows:

```

<terminated> ASS7_Q9 [Java Application] C:\Users\student\.p2\pool\plugins\org.
Is 121 a palindrome? true
Is -121 a palindrome? false

```

Q10. Write a Java program that reads two float numbers and checks whether the difference between these two numbers is less than ϵ ($\epsilon < 1$).

```

package mypackage;

import java.util.Scanner;

public class ASS7_Q10 {

    public static void main(String[] args) {
        // Create a scanner for user input
        Scanner scanner = new Scanner(System.in);

        // Read two float numbers from the user
    }
}

```

```

System.out.print("Enter the first float number: ");
float num1 = scanner.nextFloat();

System.out.print("Enter the second float number: ");
float num2 = scanner.nextFloat();

// Define epsilon (threshold)
final float EPSILON = 0.0001f; // Example: very small value less than 1

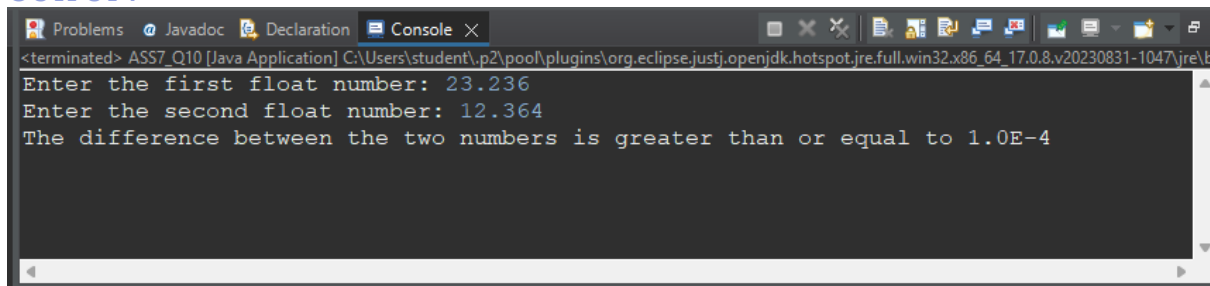
// Calculate the absolute difference
float difference = Math.abs(num1 - num2);

// Check if the difference is less than epsilon
if (difference < EPSILON) {
    System.out.println("The difference between the two numbers is less than " +
EPSILON);
} else {
    System.out.println("The difference between the two numbers is greater than or
equal to " + EPSILON);
}

// Close the scanner
scanner.close();
}
}

```

OUTPUT :-



```

<terminated> ASS7_Q10 [Java Application] C:\Users\student\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v20230831-1047\jre\bin\java.exe
Enter the first float number: 23.236
Enter the second float number: 12.364
The difference between the two numbers is greater than or equal to 1.0E-4

```

Q11. Write a Java program that reads an integer number and counts the number of digits that are even.

JAVA CODE :-

```

import java.util.Scanner;

public class ASS7_Q11 {

    public static void main(String[] args) {
        // Create a scanner for user input
        Scanner scanner = new Scanner(System.in);
    }
}

```

```

// Read an integer number from the user
System.out.print("Enter an integer: ");
int number = scanner.nextInt();

// Handle negative numbers by converting to positive
number = Math.abs(number);

int evenCount = 0; // Initialize count of even digits

// Count even digits
while (number > 0) {
    int digit = number % 10; // Extract the last digit

    // Check if the digit is even
    if (digit % 2 == 0) {
        evenCount++;
    }

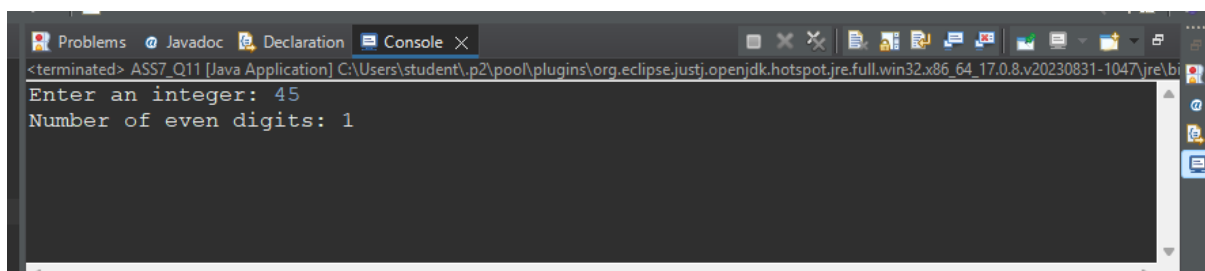
    number /= 10; // Remove the last digit
}

// Output the count of even digits
System.out.println("Number of even digits: " + evenCount);

// Close the scanner
scanner.close();
}
}

```

OUTPUT :-



```

<terminated> AS57_Q11 [Java Application] C:\Users\student\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v20230831-1047\jre\bin
Enter an integer: 45
Number of even digits: 1

```

Q12. Write a Java program that reads two integer number and create a third number by taking the first two digits of the first number and the last two digits of the second number.
Example: Input: 45678, 312 Output:4512.

JAVA CODE :-

```
import java.util.Scanner;

public class ASS7_Q12 {

    public static int createNumber(int num1, int num2) {
        // Extract the first two digits of the first number
        int firstTwoDigits = num1 / (int)Math.pow(10, (int)Math.log10(num1) - 1); // Divide
num1 by 10^(number of digits - 2)

        // Extract the last two digits of the second number
        int lastTwoDigits = num2 % 100; // Get the remainder when divided by 100

        // Combine them into a new number
        return firstTwoDigits * 100 + lastTwoDigits;
    }

    public static void main(String[] args) {
        // Create a scanner for user input
        Scanner scanner = new Scanner(System.in);

        // Read two integer numbers from the user
        System.out.print("Enter the first number: ");
        int num1 = scanner.nextInt();

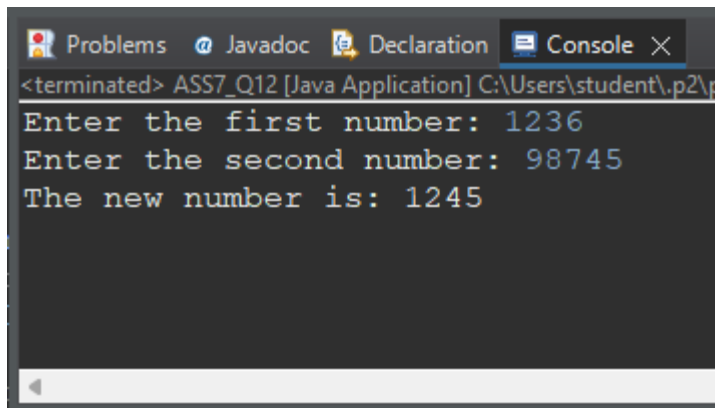
        System.out.print("Enter the second number: ");
        int num2 = scanner.nextInt();

        // Call the method to create the new number
        int result = createNumber(num1, num2);

        // Output the result
        System.out.println("The new number is: " + result);

        // Close the scanner
        scanner.close();
    }
}
```

OUTPUT :-

A screenshot of a Java IDE's console window. The window has tabs for 'Problems', 'Javadoc', 'Declaration', and 'Console'. The 'Console' tab is active, showing the output of a Java application. The text in the console is: '<terminated> ASS7_Q12 [Java Application] C:\Users\student\p2\p', 'Enter the first number: 1236', 'Enter the second number: 98745', and 'The new number is: 1245'. There is a scrollbar at the bottom of the console area.

```
<terminated> ASS7_Q12 [Java Application] C:\Users\student\p2\p
Enter the first number: 1236
Enter the second number: 98745
The new number is: 1245
```

Q13. Write a Java program to count the frequency of each digit of a number.

JAVA CODE

```
import java.util.Scanner;

public class ASS7_Q13 {
```

```

public static void countDigitFrequency(int number) {
    // Create an array to store frequency of each digit (0-9)
    int[] frequency = new int[10];

    // Handle negative numbers by converting to positive
    number = Math.abs(number);

    // Process each digit of the number
    while (number > 0) {
        int digit = number % 10; // Extract the last digit
        frequency[digit]++;      // Increment the frequency of that digit
        number /= 10;           // Remove the last digit
    }

    // Output the frequency of each digit
    System.out.println("Digit frequencies:");
    for (int i = 0; i < 10; i++) {
        if (frequency[i] > 0) {
            System.out.println("Digit " + i + ": " + frequency[i]);
        }
    }
}

public static void main(String[] args) {
    // Create a scanner for user input
    Scanner scanner = new Scanner(System.in);

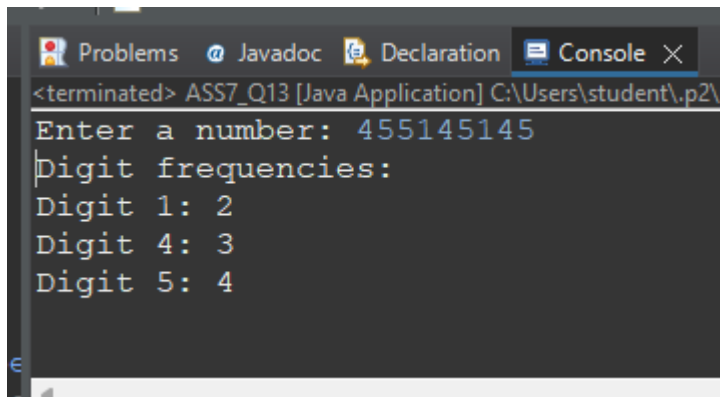
    // Read an integer number from the user
    System.out.print("Enter a number: ");
    int number = scanner.nextInt();

    // Call the method to count the digit frequency
    countDigitFrequency(number);

    // Close the scanner
    scanner.close();
}

```

OUTPUT :-



```
<terminated> ASS7_Q13 [Java Application] C:\Users\student\.p2\
Enter a number: 455145145
Digit frequencies:
Digit 1: 2
Digit 4: 3
Digit 5: 4
```

Q14. Write a Java program to check whether a number is prime or not.

JAVA CODE

```
import java.util.Scanner;

public class ASS7_Q14 {

    public static boolean isPrime(int number) {
        // Handle edge cases
        if (number <= 1) {
            return false; // Numbers less than or equal to 1 are not prime
        }

        // Check divisibility from 2 to the square root of the number
        for (int i = 2; i * i <= number; i++) {
            if (number % i == 0) {
                return false; // If divisible by any number, it's not prime
            }
        }

        return true; // If no divisors found, it's prime
    }

    public static void main(String[] args) {
        // Create a scanner for user input
        Scanner scanner = new Scanner(System.in);
```

```

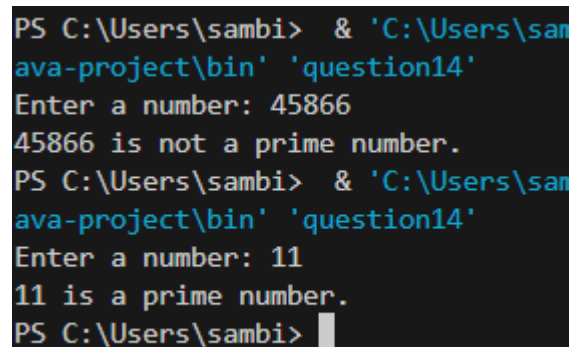
// Read an integer number from the user
System.out.print("Enter a number: ");
int number = scanner.nextInt();

// Check if the number is prime
if (isPrime(number)) {
    System.out.println(number + " is a prime number.");
} else {
    System.out.println(number + " is not a prime number.");
}

// Close the scanner
scanner.close();
}
}

```

OUTPUT



```

PS C:\Users\sambi> & 'C:\Users\sambi\Documents\Java-project\bin' 'question14'
Enter a number: 45866
45866 is not a prime number.
PS C:\Users\sambi> & 'C:\Users\sambi\Documents\Java-project\bin' 'question14'
Enter a number: 11
11 is a prime number.
PS C:\Users\sambi>

```

Q15 . Write a program to print the first 100th prime number

```

public class ASS7_Q15 {

    // Method to check if a number is prime
    public static boolean isPrime(int number) {
        if (number <= 1) {
            return false; // Numbers less than or equal to 1 are not prime
        }
        // Check divisibility from 2 to the square root of the number
    }
}

```

```

    for (int i = 2; i * i <= number; i++) {
        if (number % i == 0) {
            return false; // If divisible by any number, it's not prime
        }
    }
    return true; // If no divisors found, it's prime
}

public static void main(String[] args) {
    int count = 0; // To count prime numbers
    int number = 2; // Start checking from the number 2

    while (count < 100) { // Keep going until we find the 100th prime
        if (isPrime(number)) {
            count++; // If it's prime, increment the count
        }
        number++; // Move to the next number
    }

    // Output the 100th prime number
    System.out.println("The 100th prime number is: " + (number - 1));
}
}

```

OUTPUT :-

```

PS C:\Users\sambi> & 'C:\Users\sambi\
t.ls-java-project\bin' 'question15'
The 100th prime number is: 541
PS C:\Users\sambi>

```

Q16. Write a Java program to print the prime number in a range.

JAVA CODE :-

```

import java.util.Scanner;

public class ASS7_Q16 {

```

```

// Method to check if a number is prime
public static boolean isPrime(int number) {
    if (number <= 1) {
        return false; // Numbers less than or equal to 1 are not prime
    }
    // Check divisibility from 2 to the square root of the number
    for (int i = 2; i * i <= number; i++) {
        if (number % i == 0) {
            return false; // If divisible by any number, it's not prime
        }
    }
    return true; // If no divisors found, it's prime
}

public static void main(String[] args) {
    // Create a scanner for user input
    Scanner scanner = new Scanner(System.in);

    // Read the range from the user
    System.out.print("Enter the starting number of the range: ");
    int start = scanner.nextInt();

    System.out.print("Enter the ending number of the range: ");
    int end = scanner.nextInt();

    // Print the prime numbers in the given range
    System.out.println("Prime numbers between " + start + " and " + end + " are:");
    for (int i = start; i <= end; i++) {
        if (isPrime(i)) {
            System.out.print(i + " ");
        }
    }

    // Close the scanner
    scanner.close();
}
}

```

OUTPUT :-

```

t.ls-java-project\bin' 'question16'
Enter the starting number of the range: 1
Enter the ending number of the range: 100
Prime numbers between 1 and 100 are:
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
PS C:\Users\sambi>

```

Q17. . Write a java program to find the minimum and maximum element in an array.

JAVA CODE :-

```
import java.util.Scanner;

public class ASS7_Q18 {

    // Method to check if a number is even
    public static boolean isEven(int number) {
        // Return true if the number is divisible by 2, otherwise return false
        return number % 2 == 0;
    }

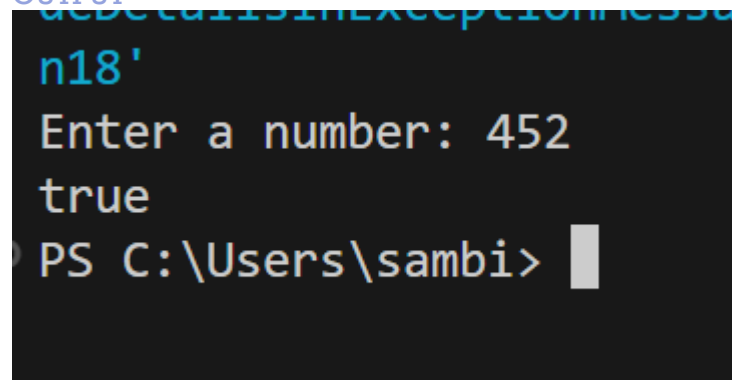
    public static void main(String[] args) {
        // Create a scanner for user input
        Scanner scanner = new Scanner(System.in);

        // Read an integer number from the user
        System.out.print("Enter a number: ");
        int number = scanner.nextInt();

        // Call the method to check if the number is even and print the result
        System.out.println(isEven(number));

        // Close the scanner
        scanner.close();
    }
}
```

OUTPUT



```
Exception in thread 'main'
n18'
Enter a number: 452
true
PS C:\Users\sambi>
```