Government College of Engineering, Jalgaon
(An Autonomous Institute of Govt. of Maharashtra)

Name: **PRN:**

Class: L.Y **Semester:** VII **Batch:**

Date of Performance: _____ **Date of Completion:** _____

Subject: CO407U CNSL **Subject Teacher:** Ms. Shrutika Mahajan

Practical – 3

Aim: Write a program that contains a string (char pointer) with a value \Hello World‘. The program should AND, OR and XOR each character in this string with 127 and display the result.

Theory:

Bitwise Operations:

In Java, bitwise operators allow manipulation of data at the binary level. Each character in Java corresponds to a Unicode/ASCII value, and that value can be represented in binary.

The number 127 (01111111_2) is chosen because it represents the 7-bit ASCII limit. Applying bitwise operations with 127 helps us study how the character bits are affected.

1. **Bitwise AND (&)** ○ Compares two bits. Result is 1 if both are 1, otherwise 0. ○ Example:
   - 72 (H) → 01001000

   127 01111111

   Result 01001000 = 72

   - Preserves the lower 7 bits of the character.

2. **Bitwise OR (|)** ○ Compares two bits. Result is 1 if either bit is 1. ○ Example:
   - 72 (H) → 01001000

   127 01111111

Result 01111111 = 127

- Forces the lower 7 bits to 1, often making the result 127.

3. **Bitwise XOR (^)** ○ Compares two bits. Result is 1 if bits are different, else 0. ○ Example:
   - ○ 72 (H) → 01001000

127 01111111

Result 00110111 = 55

- Produces the "complement" of the lower 7 bits.

Applications:

- Cryptography (XOR for simple encryption).
- Masking (AND with 127 keeps ASCII safe).
- Bit manipulation in hardware programming.

Algorithm:

1. Start program.
2. Define a string with value "Hello World". 3. Loop through each character in the string.
3. Convert character to ASCII value
5. Perform: o AND with 127 o
6. Print the results in tabular form.
7. End program.

Code:

```
public class BitwiseOperations {

   public static void AndOperation(String s) {
      System.out.print("AND : [ ");
      for (int i = 0; i < s.length(); i++) {
         int val = s.charAt(i) & 127;
         System.out.print(val + " ");
      }
      System.out.print("] Characters : ");
      for (int i = 0; i < s.length(); i++) {
         int val = s.charAt(i) & 127;
         System.out.print((char) val);
}
      System.out.println();
   }

   public static void OrOperation(String s) {
      System.out.print("OR  : [ ");
      for (int i = 0; i < s.length(); i++) {
         int val = s.charAt(i) | 127;
         System.out.print(val + " ");
      }
      System.out.print("] Characters : ");
      for (int i = 0; i < s.length(); i++) {
         int val = s.charAt(i) | 127;
         System.out.print((char) val);
      }
      System.out.println();
   }

   public static void XorOperation(String s) {
      System.out.print("XOR : [ ");
      for (int i = 0; i < s.length(); i++) {
         int val = s.charAt(i) ^ 127;
         System.out.print(val + " ");
      }
      System.out.print("] Characters : ");
      for (int i = 0; i < s.length(); i++) {
         int val = s.charAt(i) ^ 127;
         System.out.print((char) val);
      }
      System.out.println();
   }
```

```
    public static void main(String[] args) {
        String string = "Rahul Bhai";
        AndOperation(string);
        OrOperation(string);
        XorOperation(string);
    }
}
```

Output:

```
D:\Me\College\Practicals\Sem-7\CNSL>java BitwiseOperations
AND : [ 72 101 108 108 111 32 87 111 114 108 100 ] Characters : Hello World
OR  : [ 127 127 127 127 127 127 127 127 127 127 127 ] Characters :
XOR : [ 55 26 19 19 16 95 40 16 13 19 27 ] Characters : 7_(
```

Result:

The program successfully performs AND, OR, and XOR with 127 on each character of "Hello World" and displays the results.

Conclusion:

This experiment shows the use of bitwise operations on string characters:

- **AND with 127** → Preserves ASCII values (since within 7 bits).
- **OR with 127** → Converts result to 127 (since all lower 7 bits become 1).
- **XOR with 127** → Produces the complementary binary value.

This demonstrates how data can be transformed at the bit level in Java.

Course Teacher
Ms. Shrutika Mahajan