UNIVERSITY INSTITUTE OF ENGINEERING

Department of Computer Science & Engineering

(BE-CSE/IT-6th Sem)

Subject Name: Advanced Programming Lab-2

Subject Code: 22CSP-351

Submitted to:	**Submitted by:**
Er. Radha(E16496)	Name: Ishan Ojha
	UID:22BCS13921
	Section: 22BCS_IOT-617
	Group: A

PROBLEM_1:

Longest Nice Substring (Leetcode)

PROGRAM/CODE:

```
class Solution {
    public String longestNiceSubstring(String s) {
        Set<Character> set = new HashSet<>();
        for (int i = 0; i < s.length(); i++) {
            set.add(s.charAt(i));
        }
        for (int i = 0; i < s.length(); i++) {
            if (set.contains(Character.toUpperCase(s.charAt(i))) &&
                    set.contains(Character.toLowerCase(s.charAt(i)))) {
                continue;
            }
            String s1 = longestNiceSubstring(s.substring(0, i));
            String s2 = longestNiceSubstring(s.substring(i+1));
            return s1.length()>= s2.length() ? s1 : s2;
        }
        return s;
    }
}
```

PROBLEM_2:

Reverse Bits (Leetcode)

PROGRAM/CODE:

```
public class Solution {
    // you need treat n as an unsigned value
    public int reverseBits(int n) {
        int result =0;
        for (int i=0; i<32; i++) {
            result = (result << 1) ;
            result += (n & 1);
            n >>= 1;
        }
        return result;
    }
}
```

PROBLEM_3:

Number of 1 Bits (Leetcode)

PROGRAM/CODE:

```
public class Solution {
    public int hammingWeight(int n) {
        int count = 0;
        while (n != 0) {
            count += n & 1;
            n = n >>> 1;
        }
        return count;
    }
}
```

PROBLEM_4:

Maximum Subarray (Leetcode)

PROGRAM/CODE:

```
class Solution {
    public int maxSubArray(int[] nums) {
        int currentSum = nums[0];
        int maxSum = nums[0];
        for (int i = 1; i < nums.length; i++) {
            currentSum = Math.max(nums[i], currentSum + nums[i]);
            maxSum = Math.max(maxSum, currentSum);
        }
        return maxSum;
    }
}
```

PROBLEM_5:

Search a 2D Matrix II (Leetcode)

PROGRAM/CODE:

```
class Solution {
    public boolean searchMatrix(int[][] matrix, int target) {
        if (matrix == null || matrix.length == 0 || matrix[0].length == 0) {
            return false;
        }
        int row = 0;
        int col = matrix[0].length - 1;
        while (row < matrix.length && col >= 0) {
            if (matrix[row][col] == target) {
                return true;
            } else if (matrix[row][col] > target) {
                col--; // Move left
            } else {
                row++; // Move down
            }
        }
        return false;
    }
}
```

PROBLEM_6:

Super Pow (Leetcode)

PROGRAM/CODE:

```
class Solution {
    private static final int MOD = 1337;
    private int pow(int a, int b) {
        int result = 1;
        a %= MOD;  // Taking mod to prevent overflow
        for (int i = 0; i < b; i++) {
            result = (result * a) % MOD;
        }
        return result;
    }
    public int superPow(int a, int[] b) {
        int result = 1;
        for (int i = b.length - 1; i >= 0; i--) {
            result = (result * pow(a, b[i])) % MOD;
            a = pow(a, 10);  // Power up for the next iteration
        }
        return result; } }
```

PROBLEM_7:

Beautiful Array (Leetcode)

PROGRAM/CODE:

```
import java.util.ArrayList;

import java.util.List;

class Solution {

    public int[] beautifulArray(int n) {

        List<Integer> result = new ArrayList<>();

        result.add(1);

        while (result.size() < n) {

            List<Integer> temp = new ArrayList<>();

            for (int x : result) {

                if (x * 2 - 1 <= n) {

                    temp.add(x * 2 - 1); // Odd numbers

                }

            }

            for (int x : result) {

                if (x * 2 <= n) {

                    temp.add(x * 2); // Even numbers

                }

            }

            result = temp;

        }

        return result.stream().mapToInt(i -> i).toArray();

    }

}
```