**NAME:** ADITYA JETHAN

**ROLL NO:** 21BCP166

**DIV-**3 **G-**5

**DAA :** EXTRA LAB ASSIGNMENT 2

**AIM:** To make a program for binary counter for N number entered by user.

**Theory:** The program takes an input integer n and then generates all possible binary numbers with n bits by incrementing the counter n times. The output of the program is a list of n binary numbers, each separated by a newline. The program uses a vector counter of size n to store the current binary number, and increments the counter by updating the values stored in counter.

**CODE:**

#include <bits/stdc++.h>

using namespace std;

// binary counter function

void binarycounter(int n)

{

    //take bits = log2(n) + 1

    int bits = log2(n) + 1;

    vector<int> counter(bits);

    // print the initial counter

    for (int i = 0; i < bits; i++)

    {

        cout << counter[i];

    }

    cout << endl;

    // increment the counter

    for (int i = 0; i <=n; i++)

    {

        int j = bits - 1;

        while (j >= 0)

        {

            if (counter[j] == 0)

            {

                counter[j] = 1;

                break;

            }

            else

            {

                counter[j] = 0;

                j--;

            }

        }

        // print the counter

        for (int i = 0; i < bits; i++)

        {

            cout << counter[i];

        }

        cout << endl;

    }

}

// main function

int main()

{

    int n;

    cout << "Enter the number of times you want to increase the counter: ";

    cin >> n;

    // call the binary counter function

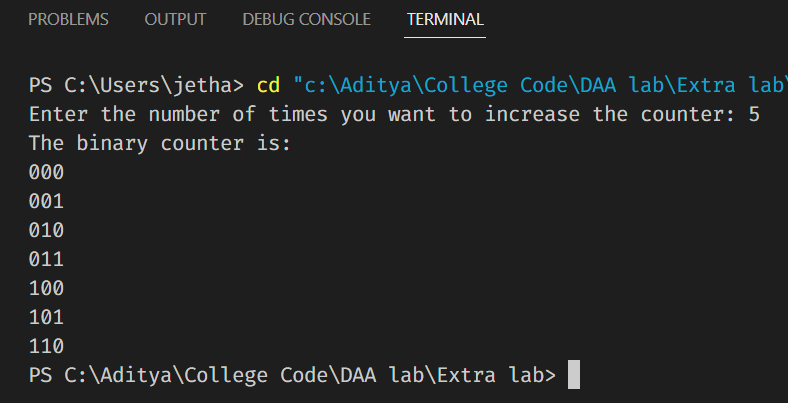
    cout << "The binary counter is: " << endl;

    binarycounter(n);

    return 0;

}

**OUTPUT:**



**ANALYSIS:**

**WORST CASE**

The worst case scenario for this code occurs when the input n is at its maximum value. In this case, the code will generate 2^n binary numbers and output them to the console. This means the time complexity of the code will be O(2^n \* n^2), which is exponential. In the worst case, when n is large, this code will take a long time to run.

**AMORTIZE ANALYSIS:**

In this code, the binary counter function is called once with an input of n, and generates and outputs 2^n binary numbers. The time complexity of each iteration of the inner for loop is O(n), and it runs 2^n times, so the total time complexity of the inner loop is O(n \* 2^n).

Since the outer loop only runs n times and the time complexity of each iteration is O(n \* 2^n), the total time complexity of the code can be expressed as O(n \* (n \* 2^n)) = O(n^2 \* 2^n).

Therefore, the amortized time complexity of this code is O(n^2 \* 2^n), which is exponential.