Experiment 10

Student Name: Abhishek Roushan **UID:** 22BCS10187
Branch: BE-CSE **Section/Group:** 22BCS_NTPP_IOT_603
Semester: 6th **Date of Performance:** 18-04-25
Subject Name: AP Lab-2 **Subject Code:** 22CSP-351

1. Aim: Pascal's Triangle

2. Objective:

Given an integer numRows, return the first numRows of **Pascal's triangle**.

3. Implementation/Code:

```
vector<vector<int>> generate(int numRows) {

        vector<vector<int>> a(numRows);

        for(int&& i=0; i<numRows; i++){

            a[i].assign(i+1, 1);

            for(int&& j=1; j<=i/2; j++){

                  a[i][i-j]=a[i][j]=a[i-1][j-1]+a[i-1][j];

            }
        }
        return a;
```

4. Output

Input
numRows = 5

Output
[[1],[1,1],[1,2,1],[1,3,3,1],[1,4,6,4,1]]

Expected
[[1],[1,1],[1,2,1],[1,3,3,1],[1,4,6,4,1]]

Input
numRows = 1

Output
[[1]]

Expected
[[1]]

Question 2

1. Aim:- **Trapping Rain Water**

2. Objective:-

Given n non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it can trap after raining.

3. Implementation/Code:-

```
class Solution {
public:

    int trap(vector<int>& height) {

        int l=0;
        int r=height.size()-1;
        int lmax=INT_MIN;
        int rmax=INT_MIN;
        int ans=0;

        while(l<r){
            lmax=max(lmax,height[l]);
            rmax=max(rmax,height[r]);
            ans+=(lmax<rmax)?lmax-height[l++]:rmax-height[r--];
        }
        return ans;
    }
};
```

4. Output:-

Input
height = [0,1,0,2,1,0,1,3,2,1,2,1]
Output
6
Expected
6

Input
height = [4,2,0,3,2,5]
Output
9
Expected
9

Question 3

5. **Aim**:- **Valid Palindrome**

6. **Objective**:-

A phrase is a **palindrome** if, after converting all uppercase letters into lowercase letters and removing all non-alphanumeric characters, it reads the same forward and backward. Alphanumeric characters include letters and numbers.

Given a string s, return true *if it is a* ***palindrome****, or* false *otherwise*.

7. **Implementation/Code:-**

```
class Solution {
public:

bool help(string &str){

    int i = 0;
    int j = str.size()-1;

    while(i<=j){
        if(str[i] != str[j]){
            return false;
        }
        i++;
        j--;
    }
    return true;
}
    bool isPalindrome(string s) {

        if(s.empty()){
            return true;
        }

        string str = "";

        for(int i=0;i<s.size();i++){
            if((s[i] >= 'a' and s[i] <= 'z') ){
                str.push_back(s[i]);
            }else{
                if(s[i] >= 'A' and s[i] <= 'Z'){
```

```
                    str.push_back(s[i]+32);
                }else{
                    if(s[i] >= '0' and s[i] <= '9'){
                        str.push_back(s[i]);
                    }
                }
            }
        }
      //  cout<<str<<" ";

        return help(str);

    }
};
```

8. Output:-

Input
s = "A man, a plan, a canal: Panama"
Output
true
Expected
true

Question 4

9. Aim:- Trapping Rain Water

10. Objective:-

Given an m x n matrix where each row is sorted in ascending order from left to right and each column is sorted in ascending order from top to bottom, and an integer target, determine if the target exists in the matrix.

11. Implementation/Code:-

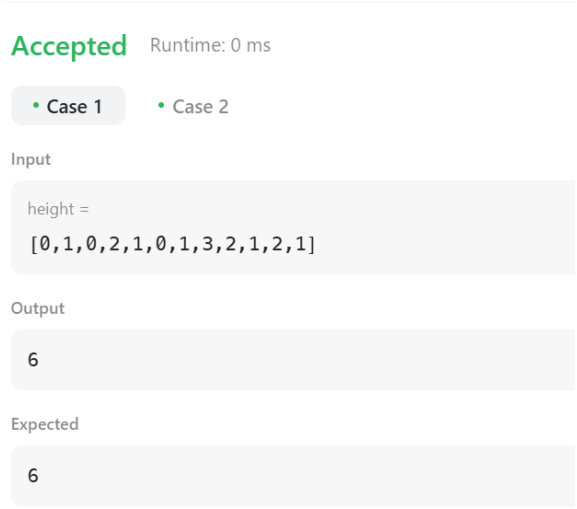
```
class Solution {
public:

    int trap(vector<int>& height) {

        int l=0;
        int r=height.size()-1;
        int lmax=INT_MIN;
        int rmax=INT_MIN;
        int ans=0;

        while(l<r){
            lmax=max(lmax,height[l]);
            rmax=max(rmax,height[r]);
            ans+=(lmax<rmax)?lmax-height[l++]:rmax-height[r--];
        }
        return ans;
    }
};
```

12. Output:-



13.Learning Outcome:

- We learn about to traverse in 0(n).
- We learn about while loop function .

Question 5

14.Aim:- **Word Break**

15.Objective:-

Given a string s and a dictionary wordDict containing a list of words, determine if s can be segmented into a space-separated sequence of one or more dictionary words. The same word can be reused multiple times.

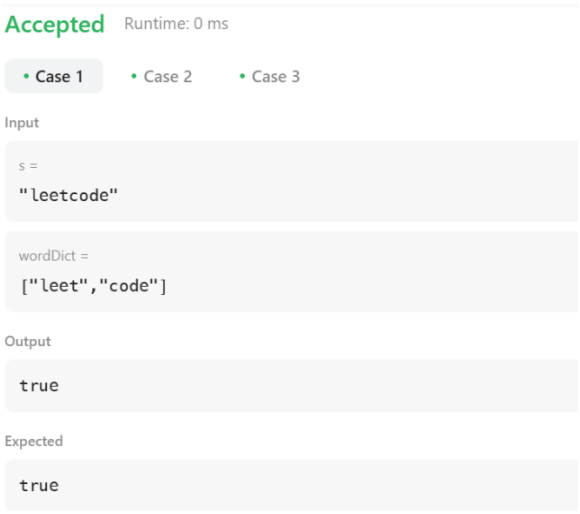
16.Implementation/Code:-

```
class Solution {
public:
    bool wordBreak(string s, vector<string>& wordDict) {

        unordered_set<string> wordSet(wordDict.begin(), wordDict.end());
        vector<bool> dp(s.length() + 1, false);
        dp[0] = true;

        for (int i = 1; i <= s.length(); i++) {
            for (int j = 0; j < i; j++) {
                if (dp[j] && wordSet.find(s.substr(j, i - j)) !=
wordSet.end()) {
                    dp[i] = true;
                    break;
                }
            }
        }
        return dp[s.length()];
    }
};
```

17. Output:-



18.Learning Outcome:

- We learn about to Unordered_Set.
- We learn about DP .