

## WORKSHEET 9

Student Name: Manpreet Kaur

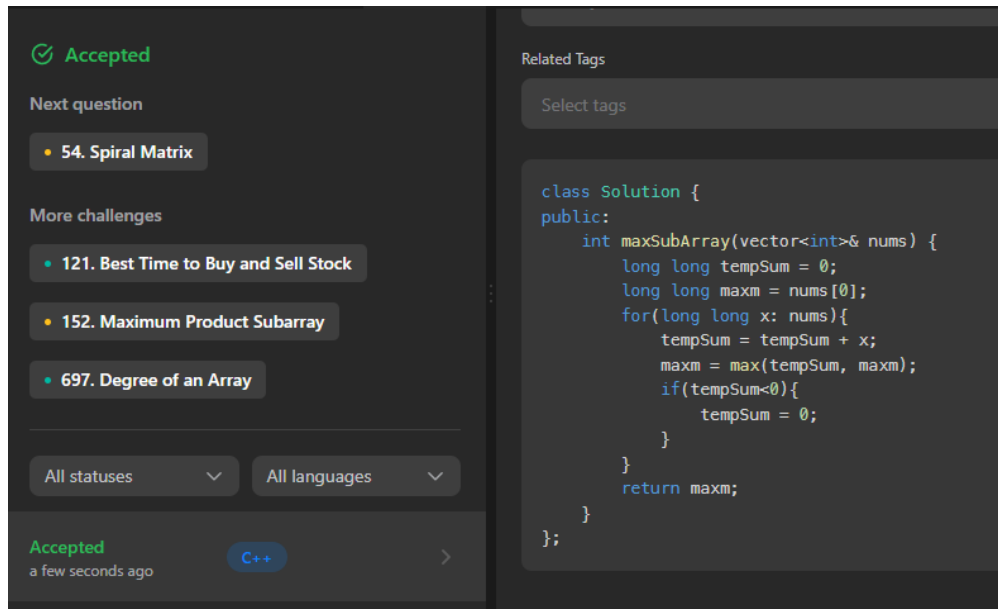
DOMAIN CAMP: 03-01-2023 to 14-01-2023

Subject Name: IT Skills (DSA)

UID: 20BCS2212

Section/Group: DWWC-43

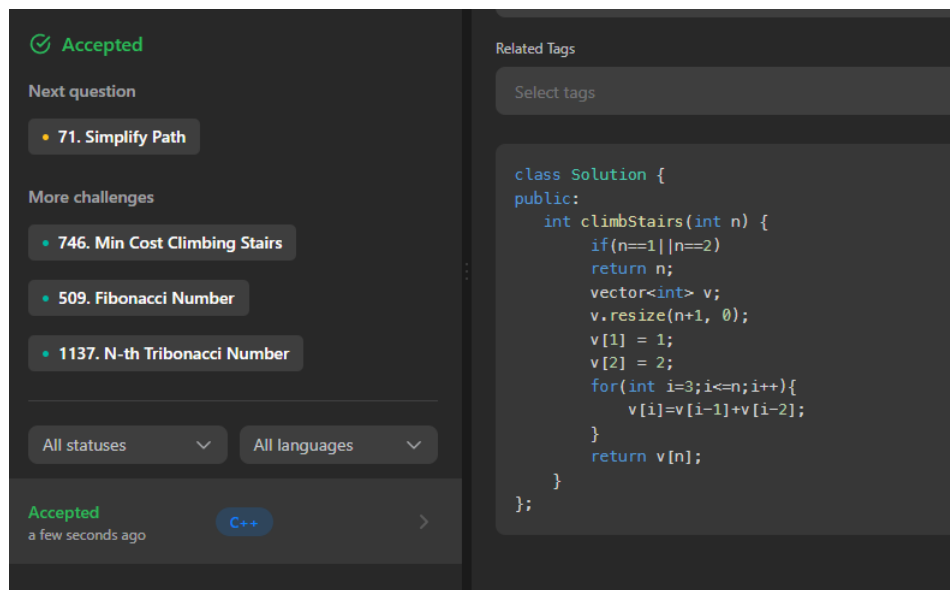
### Question 1. MAXIMUM SUBARRAY



The screenshot shows the LeetCode interface for the "Maximum Subarray" problem. On the left, it indicates the solution is "Accepted" and lists "Next question" (54. Spiral Matrix) and "More challenges" (121. Best Time to Buy and Sell Stock, 152. Maximum Product Subarray, 697. Degree of an Array). On the right, the C++ code for the solution is displayed:

```
class Solution {
public:
    int maxSubArray(vector<int>& nums) {
        long long tempSum = 0;
        long long maxm = nums[0];
        for(long long x: nums){
            tempSum = tempSum + x;
            maxm = max(tempSum, maxm);
            if(tempSum<0){
                tempSum = 0;
            }
        }
        return maxm;
    }
};
```

### Question 2. CLIMBING STAIRS



The screenshot shows the LeetCode interface for the "Climbing Stairs" problem. On the left, it indicates the solution is "Accepted" and lists "Next question" (71. Simplify Path) and "More challenges" (746. Min Cost Climbing Stairs, 509. Fibonacci Number, 1137. N-th Tribonacci Number). On the right, the C++ code for the solution is displayed:

```
class Solution {
public:
    int climbStairs(int n) {
        if(n==1||n==2)
            return n;
        vector<int> v;
        v.resize(n+1, 0);
        v[1] = 1;
        v[2] = 2;
        for(int i=3;i<=n;i++){
            v[i]=v[i-1]+v[i-2];
        }
        return v[n];
    }
};
```

### Question 3. PASCAL'S TRIANGLE

Accepted

Next question

119. Pascal's Triangle II

More challenges

119. Pascal's Triangle II

All statuses

All languages

Accepted

a few seconds ago

C++

```

class Solution {
public:
    vector<vector<int>>> generate(int num) {
        vector<vector<int>>> ans;
        for(int i=0;i<num;i++){
            vector<int> temp(i+1,1);
            if(i<=1){
                ans.push_back(temp);
            }
            else{
                for(int j=1;j<=i-1;j++){
                    temp[j]=ans[i-1][j-1]+ans[i-1][j];
                }
                ans.push_back(temp);
            }
        }
        return ans;
    }
};

```

### Question 4. LONGEST PALINDROMIC SUBSTRING

Accepted

Next question

6. Zigzag Conversion

More challenges

214. Shortest Palindrome

266. Palindrome Permutation

336. Palindrome Pairs

All statuses

All languages

Accepted

a few seconds ago

C++

```

class Solution {
public:
    int check(string &s, int L, int R)
    {
        while(L>=0 and R<s.length() and s[L]==s[R])
        {
            L--;
            R++;
        }
        return R-L-1;
    }

    string longestPalindrome(string s) {
        int ans = 0, st=0;
        int n = s.length();

        for(int i = 0;i<n;i++)
        {
            int len1 = check(s, i,i);
            int len2 = check(s, i, i+1);

            int len = max(len1, len2);

            if(len> ans)
            {
                ans = len;
                st = i-(len-1)/2;
            }
        }
        return s.substr(st, ans);
    }
};

```

## Question 5. FIBONACCI NUMBER

Accepted

Next question

- 973. K Closest Points to Origin

More challenges

- 842. Split Array into Fibonacci Sequence
- 873. Length of Longest Fibonacci Subsequence
- 1137. N-th Tribonacci Number

All statuses All languages

Accepted a few seconds ago C++

Runtime 11 ms Beats 46.32%

Notes

Write your notes here

Related Tags

Select tags

```

class Solution {
public:
    int fib(int n) {
        return n <= 1 ? n : fib(n - 1) + fib(n - 2);
    }
};

```

## Question 6. MAXIMUM PROFIT IN JOB SCHEDULING

Accepted

Next question

- 2273. Find Resultant Array After Removing Anagrams

More challenges

- 2008. Maximum Earnings From Taxi
- 2054. Two Best Non-Overlapping Events

All statuses All languages

Accepted a few seconds ago C++

```

class Solution {
public:
    int jobScheduling(vector<int>& startTime, vector<int>& endTime, vector<int>& profit) {
        // step 1: generate the sorting index such that endTime[index[i]] <= endTime[index[i+1]] for 0 <= i < n-1
        auto comp = [&endTime](const int i1, const int i2) { return endTime[i1] < endTime[i2]; };
        int n = endTime.size();
        vector<int> index(n);
        iota(index.begin(), index.end(), 0);
        sort(index.begin(), index.end(), comp);

        // step 2: in order to perform binary search, we also need the sorting endTime array
        vector<int> endSorted(endTime.begin(), endTime.end());
        sort(endSorted.begin(), endSorted.end());

        // step 3: calculate dp table
        vector<int> dp(n + 1);
        for (int i = 1; i <= n; i++) {
            int j = upper_bound(endSorted.begin(), endSorted.end(), startTime[index[i-1]]) - endSorted.begin();
            dp[i] = max(dp[i-1], profit[index[i-1]] + dp[j]);
        }
        return dp[n];
    }
};

```

## Question 7. UNIQUE PATHS

Accepted

Next question

- 63. Unique Paths II

More challenges

- 63. Unique Paths II
- 64. Minimum Path Sum
- 174. Dungeon Game

All statuses ▼ All languages ▼

Accepted  
a few seconds ago

C++ ➤

```

class Solution {
public:
    int nl, ml, dp[105][105];

    bool valid(int x, int y) {
        if(x < nl and y < ml) return true;
        else return false;
    }

    int solve(int i, int j) {
        if(!valid(i, j)) return 0;
        if(i >= nl-1 and j >= ml-1) return 1;
        if(dp[i][j] != -1) return dp[i][j];
        int ans = solve(i+1, j) + solve(i, j+1);
        return dp[i][j] = ans;
    }

    int uniquePaths(int m, int n) {
        nl = m, ml = n;
        memset(dp, -1, sizeof(dp));
        int ans = solve(0, 0);
        return ans;
    }
};

```

## Question 8. PERFECT SQUARES

Accepted

Next question

- 280. Wiggle Sort

More challenges

- 204. Count Primes
- 264. Ugly Number II

All statuses ▼ All languages ▼

Accepted  
a few seconds ago

C++ ➤

Related tags

Select tags

```

class Solution {
public:
    int numSquares(int n) {
        vector<int> dp(n + 1, (int) 1e9);
        int i, j;

        i = 0;
        dp[0] = 0;
        while (++i < n + 1)
        {
            j = 1;
            while (j * j < i + 1)
                dp[i] = min(dp[i], dp[i - j * j] + 1);
            j++;
        }
        return dp[n];
    }
};

```