Experiment-3.1

Student Name: UTKARSH JOSHI **UID: 21BCS9158**
Branch: CSE **Section/Group: 802-A**
Semester: 5th **Date of Performance: 07/11/23**
Subject Name: AP Lab-1 **Subject Code: 21CSP-314**

Aim:

Dynamic Programming

Dynamic Programming is mainly an optimization over plain recursion. Wherever we see a recursive solution that has repeated calls for same inputs, we can optimize it using Dynamic Programming. The idea is to simply store the results of subproblems, so that we do not have to re-compute them when needed later. This simple optimization reduces time complexities from exponential to polynomial.

TASK-1:

Construct the Array

TASK-2:

Sam and substrings

Objective:

To understand the concept of dynamic programming.

Source Code:

TASK-1:

```
#include <bits/stdc++.h>

using namespace std;

const int MOD = 1000000007;

void print(long arr[2][2]) {   for
(int i = 0; i < 2; i++) {     for
(int j = 0; j < 2; j++) {
     cout << i << ' ' << j << ' ' << arr[i][j] << endl;     }
  }
```

```
}



long countArray(int n, int k, int x) {
long ways[2][2];   ways[0][0] = 1;
ways[0][1] = 0;   bool fillSecond = true;
for (int i = 0; i < n-1; i++) {
   ways[fillSecond][0] = (ways[!fillSecond][1] * (k - 1)) %  MOD;
   ways[fillSecond][1] = (ways[!fillSecond][1] * (k - 2) + ways[!fillSecond][0]) %  MOD;
   fillSecond = !fillSecond;
 }

 long answer;
if (x == 1) {
   answer = (ways[fillSecond][1] * (k - 1)) % MOD;
 }   else
{
   answer = (ways[fillSecond][1] * (k - 2) + ways[fillSecond][0]) % MOD;
 }
 return answer;
}

int main() {
int n;     int k;
int x;
   cin >> n >> k >> x;
   long answer = countArray(n, k, x);
cout << answer << endl;
   return 0;
}
```

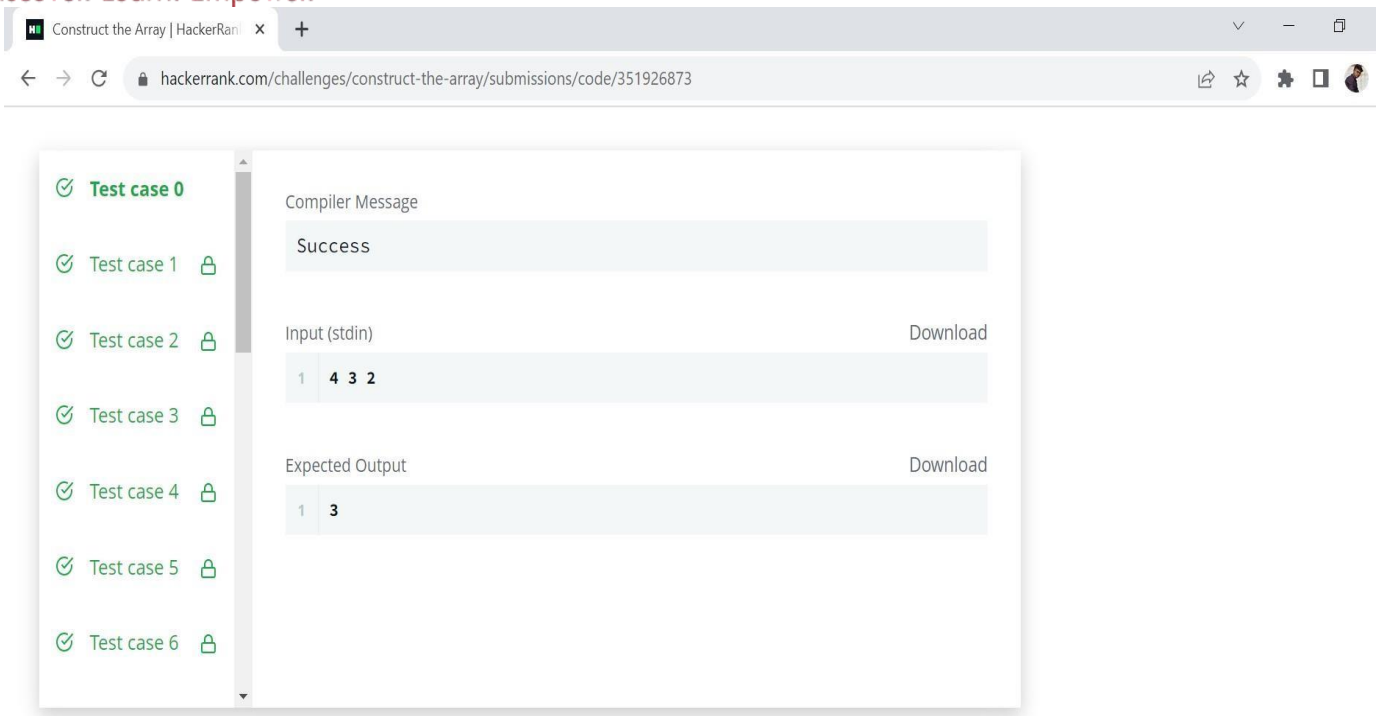
TASK-2:

```
#include    <bits/stdc++.h> using
namespace std; long
long mod = 1000000007;
```

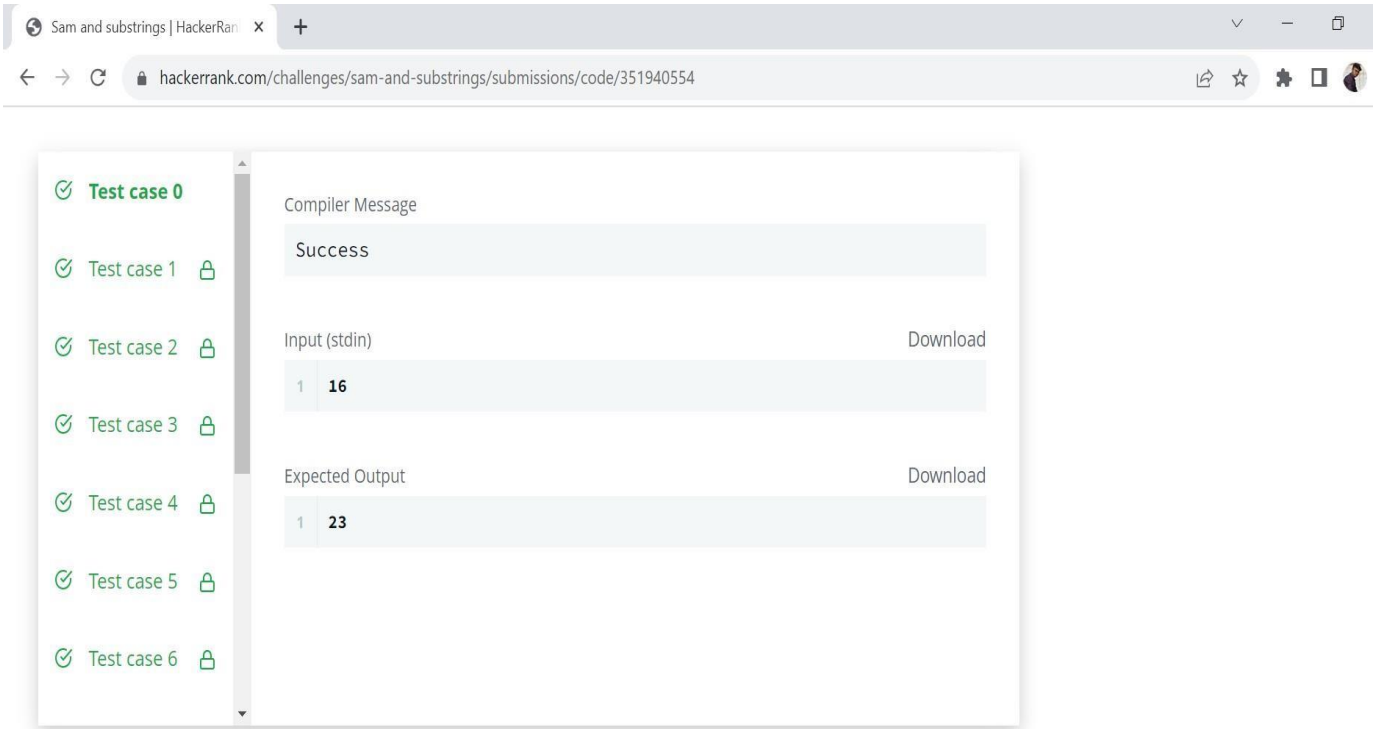
```
int main() {
string s;
   cin >> s;


long long pw[s.length() + 1]; pw[0]
= 1;
   for (int i = 1; i <= (int)s.length(); ++i) {
pw[i] = pw[i - 1] * 10 % mod;
   }
   long long sum[s.length() + 1];
sum[0] = pw[0];
   for (int i = 1; i <= (int)s.length(); ++i) {
sum[i] = sum[i - 1] + pw[i];        sum[i]
%= mod;
   }      long long ans = 0;
int n
= s.length();     for (int i =
0; i < n; ++i) {          ans += (s[i] - '0') * (i + 1)
* sum[n - 1 - i];         ans %= mod;
   }
   cout << ans << endl;
return 0;
}
```

Output:

TASK-1:


Construct the Array | HackerRan
hackerrank.com/challenges/construct-the-array/submissions/code/351926873
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Input (stdin)
Download
1 4 3 2
Expected Output
Download
1 3

TASK-2:


Sam and substrings | HackerRan
hackerrank.com/challenges/sam-and-substrings/submissions/code/351940554
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Input (stdin)
Download
1 16
Expected Output
Download
1 23

Learning Outcomes:

- Understood the concept of dynamic programming.
- It is mainly an optimization over plain recursion.
- The final general characteristic of the dynamic-programming approach is the development of a recursive optimization procedure, which builds to a solution of the overall N-stage problem by first solving a one-stage problem.