Experiment 3.1

Student Name: Nishant Kumar Mehta **UID: 21BCS3402**
Branch: CSE **Section/Group: IOT - 602**
Semester: 5th **Dateof Performance: 20/09/23**
Subject Name: Advance Programming Lab **Subject Code: 21CSP - 314**

1. Aim:
Implement the problems based on Dynamic Programming.

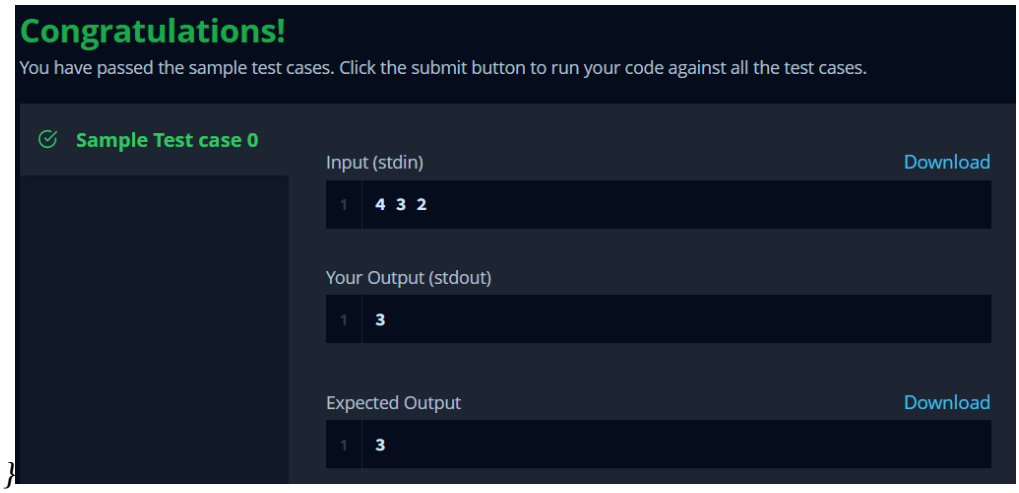
2. Objective:

- Your goal is to find the number of ways to construct an array such that consecutive positions contain different values.Specifically, we want to construct an array with n elements such that each element between 1 and k, inclusive. We also want the first and last elements of the array to be 1 and x.Given n, and x,find the number of ways to construct such an array. Since the answer may be large, only find it modulo 10^9 + 7.
- Christy is interning at HackerRank. One day she has to distribute some chocolates to her colleagues. She is biased towards her friends and plans to give them more than the others. One of the program managers hears of this and tells her to make sure everyone gets the same number.To make things difficult, she must equalize the number of chocolates in a series of operations. For each operation, she can give 1,2 or 5 pieces to all but one colleague. Everyone who gets a piece in a round receives the same number of pieces.Given a starting distribution, calculate the minimum number of operations needed so that every colleague has the same number of pieces..

3. Program and output:

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;
public class Solution {
    static long countArray(int n, int k, int x) {
```

```
    // Return the number of ways to fill in the array.
    long max = 1000000007;
    long[] f = new long[n + 1]; //[1,...,1]
    long[] g = new long[n + 1]; //[1,...,2]
    f[3] = k - 1;
    g[2] = 1;
    g[3] = k - 2;
    for (int i = 4; i <= n; i++) {
        f[i] = (k - 1) * g[i - 1] % max;
        g[i] = (f[i - 1] + (k - 2) * g[i - 1]) % max;
    }
    return x == 1 ? f[n] : g[n];
}
public static void main(String[] args) {
    Scanner in = new Scanner(System.in);
    int n = in.nextInt();
    int k = in.nextInt();
    int x = in.nextInt();
    long answer = countArray(n, k, x);
    System.out.println(answer);
    in.close();
}
```



}

2.

```
import java.io.BufferedInputStream;
import java.util.Scanner;
public class Solution {
    static int offset = 100;
    static Integer f[][] = new Integer[1111][1111];
    static int a[] = new int[11111];
    // From i down to j
    static Integer F(int i, int j) {
        if (i < j) return Integer.MAX_VALUE / 2;
        if (f[i + offset][j + offset] != null) return f[i + offset][j + offset];
        if (i == j) return 0;
        int ans = Integer.MAX_VALUE / 2;
        ans = Math.min(ans, F(i - 1, j) + 1);
        ans = Math.min(ans, F(i - 2, j) + 1);
        ans = Math.min(ans, F(i - 5, j) + 1);
        return f[i + offset][j + offset] = ans;
    }
    public static void main(String[] args) {
        Scanner cin = new Scanner(new BufferedInputStream(System.in));
        for (int T = cin.nextInt(); T!=0; T--) {
            int n = cin.nextInt();
            int min = Integer.MAX_VALUE;
            for (int i=0; i<n; i++) {
                a[i] = cin.nextInt();
```

```
            min = Math.min(min, a[i]);
        }
        int ans = Integer.MAX_VALUE;
        for (int i=min; i>=min-30; i--) {
            int tmp = 0;
            for (int j=0; j<n; j++) {
                tmp += F(a[j], i);
            }
            ans = Math.min(ans, tmp);
        }
        System.out.println(ans);
    }
}
```

Learning Outcomes:

- Learnt about Dynamic Programming.
- Learnt about how to optimize a Problem using DP.
- Learnt about Dynamic approach to solve a problem.