

**NAME: AKSHAYA B**

**REG NO: 241801012**

## **DYNAMIC PROGRAMMING**

### **PROGRAM 1:**

Question 1 | Correct Mark 10.00 out of 10.00 [Flag question](#)

#### **Playing with Numbers:**

Ram and Sita are playing with numbers by giving puzzles to each other. Now it was Ram term, so he gave Sita a positive integer 'n' and two numbers 1 and 3. He asked her to find the possible ways by which the number n can be represented using 1 and 3. Write any efficient algorithm to find the possible ways.

#### **Example 1:**

**Input:** 6

**Output:** 6

**Explanation:** There are 6 ways to 6 represent number with 1 and 3

1+1+1+1+1+1

3+3

1+1+1+3

1+1+3+1

1+3+1+1

3+1+1+1

#### **Input Format**

First Line contains the number n

#### **Output Format**

**Print:** The number of possible ways 'n' can be represented using 1 and 3

Sample Input

6

Sample Output

6

**Answer:** (penalty regime: 0 %)

```
1 #include <stdio.h>
2
3 long long countWays(int n) {
4     if (n < 0) return 0;
5     if (n == 0) return 1;
6
7     long long dp[n + 1];
8     dp[0] = 1;
9
10    for (int i = 1; i <= n; i++) {
11        dp[i] = 0;
12        if (i - 1 >= 0)
13            dp[i] += dp[i - 1];
14        if (i - 3 >= 0)
15            dp[i] += dp[i - 3];
16    }
17
18    return dp[n];
19 }
20
21 int main() {
22     int n;
23     scanf("%d", &n);
24     printf("%lld", countWays(n));
25     return 0;
26 }
27
```

|   | Input | Expected          | Got               |   |
|---|-------|-------------------|-------------------|---|
| ✓ | 6     | 6                 | 6                 | ✓ |
| ✓ | 25    | 8641              | 8641              | ✓ |
| ✓ | 100   | 24382819596721629 | 24382819596721629 | ✓ |

Passed all tests! ✓

## PROGRAM 2:

Question 1 | Correct Mark 10.00 out of 10.00 [Flag question](#)

### Playing with Chessboard:

Ram is given with an  $n \times n$  chessboard with each cell with a monetary value. Ram stands at the (0,0), that is the position of the top left white rook. He is given a task to reach the bottom right black rook position ( $n-1$ ,  $n-1$ ) constrained that he needs to reach the position by traveling the maximum monetary path under the condition that he can only travel one step right or one step down the board. Help Ram to achieve it by providing an efficient DP algorithm.

### Example:

#### Input

```
3
1 2 4
2 3 4
8 7 1
```

#### Output:

```
19
```

### Explanation:

Totally there will be 6 paths among that the optimal is

Optimal path value:  $1+2+8+7+1=19$

### Input Format

First Line contains the integer  $n$

The next  $n$  lines contain the  $n \times n$  chessboard values

### Output Format

Print Maximum monetary value of the path

Answer: (penalty regime: 0 %)

```
1 #include <stdio.h>
2 int max(int a, int b)
3 {
4     return (a > b) ? a : b;
5 }
6 int main()
7 {
8     int n;
9     scanf("%d", &n);
10    int board[n][n];
11    int dp[n][n];
12    for (int i = 0; i < n; i++)
13        for (int j = 0; j < n; j++)
14        {
15            scanf("%d", &board[i][j]);
16            dp[i][j] = board[i][j];
17        }
18    for (int j = 1; j < n; j++)
19        dp[0][j] = dp[0][j-1] + board[0][j];
20    for (int i = 1; i < n; i++)
21        dp[i][0] = dp[i-1][0] + board[i][0];
22    for (int i = 1; i < n; i++)
23        for (int j = 1; j < n; j++)
24        {
25            dp[i][j] = board[i][j] + max(dp[i-1][j], dp[i][j-1]);
26        }
27    printf("%d\n", dp[n-1][n-1]);
28    return 0;
29 }
30
```

|   | Input   | Expected | Got |   |
|---|---|----------|-----|---|
| ✓ | 3<br>1 2 4<br>2 3 4<br>8 7 1                  | 19       | 19  | ✓ |
| ✓ | 3<br>1 3 1<br>1 5 1<br>4 2 1                  | 12       | 12  | ✓ |
| ✓ | 4<br>1 1 3 4<br>1 5 7 8<br>2 3 4 6<br>1 6 9 0 | 28       | 28  | ✓ |

Passed all tests! ✓

**Correct**

Marks for this submission: 10.00/10.00.

## PROGRAM 3:

**Question 1** | Correct Mark 1.00 out of 1.00 [Flag question](#)

Given two strings find the length of the common longest subsequence(need not be contiguous) between the two.

Example:

s1: ggtabe

s2: tgatasb

|    |          |   |          |          |          |          |          |
|----|----------|---|----------|----------|----------|----------|----------|
| s1 | a        | g | <b>g</b> | <b>t</b> | <b>a</b> | <b>b</b> |          |
| s2 | <b>g</b> | x | <b>t</b> | x        | <b>a</b> | y        | <b>b</b> |

**The length is 4**

Solveing it using Dynamic Programming

**For example:**

| Input | Result |
|-------|--------|
| aab   | 2      |
| azb   |        |


**Answer:** (penalty regime: 0 %)

```
1 #include <stdio.h>
2 #include <string.h>
3 int max(int a, int b)
4 {
5     return (a > b) ? a : b;
6 }
7 int main()
8 {
9     char s1[1000], s2[1000];
10    scanf("%s %s", s1, s2);
11    int m = strlen(s1);
12    int n = strlen(s2);
13    int dp[m+1][n+1];
14    for (int i = 0; i <= m; i++)
15    {
16        for (int j = 0; j <= n; j++)
17        {
18            if (i == 0 || j == 0)
19                dp[i][j] = 0;
20            else if (s1[i-1] == s2[j-1])
21                dp[i][j] = dp[i-1][j-1] + 1;
22            else
23                dp[i][j] = max(dp[i-1][j], dp[i][j-1]);
24        }
25    }
26    printf("%d\n", dp[m][n]);
27    return 0;
28 }
29
```

|   | Input        | Expected | Got |   |
|---|--------------|----------|-----|---|
| ✓ | aab<br>azb   | 2        | 2   | ✓ |
| ✓ | ABCD<br>ABCD | 4        | 4   | ✓ |

Passed all tests! ✓

## PROGRAM 4:

**Question 1** | Correct Mark 1.00 out of 1.00  [Flag question](#)

Problem statement:

Find the length of the Longest Non-decreasing Subsequence in a given Sequence.

Eg:

Input:9

Sequence:[-1,3,4,5,2,2,2,2,3]

the subsequence is [-1,2,2,2,2,3]

Output:6

**Answer:** (penalty regime: 0 %)

```
1 #include <stdio.h>
2 int max(int a, int b)
3 {
4     return (a > b) ? a : b;
5 }
6 int main()
7 {
8     int n;
9     scanf("%d", &n);
10    int seq[n], dp[n];
11    for (int i = 0; i < n; i++)
12        scanf("%d", &seq[i]);
13    for (int i = 0; i < n; i++)
14        dp[i] = 1;
15    for (int i = 1; i < n; i++)
16    {
17        for (int j = 0; j < i; j++)
18        {
19            if (seq[i] >= seq[j])
20            {
21                dp[i] = max(dp[i], dp[j] + 1);
22            }
23        }
24    }
25    int ans = 0;
26    for (int i = 0; i < n; i++)
27        if (dp[i] > ans)
28            ans = dp[i];
29    printf("%d\n", ans);
30    return 0;
31 }
32
```

|   | Input                   | Expected | Got |   |
|---|-------------------------|----------|-----|---|
| ✓ | 9<br>-1 3 4 5 2 2 2 2 3 | 6        | 6   | ✓ |
| ✓ | 7<br>1 2 2 4 5 7 6      | 6        | 6   | ✓ |

Passed all tests! ✓

**Correct**

Marks for this submission: 1.00/1.00.