ROLL NO:230701036

NAME: ARUN MC

TOPIC: DYNAMIC PROGRAMMING

1-DP-Playing with Numbers

AIM:

Write any efficient PROGRAM to find the possible ways with the integer 1 and 3.

CODE:

```
#include <stdio.h>


int main() {
   int n;


   // Input the number n
   //printf("Enter the number n: ");
   scanf("%d", &n);


   // Create a DP array to store the number of ways to represent each number
   long long dp[n + 1]; // Use long long to handle larger numbers


   // Base cases
   dp[0] = 1; // There's 1 way to represent 0 (using no numbers)


   for (int i = 1; i <= n; i++) {
      dp[i] = dp[i - 1]; // Always include the count of ways from (i - 1)
```

```
        if (i >= 3) {
            dp[i] += dp[i - 3]; // Include the count of ways from (i - 3)
        }
    }

    // Output the number of ways to represent n
    printf("%lld\n",dp[n]);

    return 0;
}
```

INPUT:

First Line contains the number n

OUTPUT:

	Input	Expected	Got	
✔	6	6	6	✔
✔	25	8641	8641	✔
✔	100	24382819596721629	24382819596721629	✔

Passed all tests! ✔

Correct

Marks for this submission: 10.00/10.00.

2-DP-Playing with chessboard

AIM:

To determine the maximum monetary value path on an n×nn \times nn×n chessboard, starting from the top-left corner (0,0)(0,0)(0,0) and ending at the bottom-right corner (n−1,n−1)(n-1, n-1)(n−1,n−1). The movement is restricted to one step right or one step down at each step.

CODE:

```
#include<stdio.h>
int main(){
   int n;
   scanf("%d",&n);
   int a[n][n],sum=0;
   for(int i=0;i<n;i++)
     for(int j=0;j<n;j++)
      scanf("%d",&a[i][j]);
   sum+=a[0][0];
   for(int i=0,j=0;i+j<2*n;)
   {

         if(i!=n-1 && j!=n-1)
         {

           if(a[i+1][j]>=a[i][j+1])
```

```
            i++;

            else if(a[i][j+1]>a[i+1][j])
            j++;

           sum+=a[i][j];
       }

       else {
           if(i==n-1&&j!=n-1)
              j++;

           else if(j==n-1&&i!=n-1)
              i++;
           else
              break;
         sum+=a[i][j];

        }
    }
    printf("%d",sum);
}
```

INPUT:

3
1 2 4

2 3 4
8 7 1

OUTPUT:

	Input	Expected	Got	
✔	3 1 2 4 2 3 4 8 7 1	19	19	✔
✔	3 1 3 1 1 5 1 4 2 1	12	12	✔
✔	4 1 1 3 4 1 5 7 8 2 3 4 6 1 6 9 0	28	28	✔

Passed all tests! ✔

Correct

Marks for this submission: 10.00/10.00.

3-DP-Longest Common Subsequence

AIM:

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

CODE:

```
    #include <stdio.h>
#include <string.h>


int main() {
   char s1[100], s2[100];
   int dp[101][101]; // DP table for LCS

   // Input two strings
   //printf("Enter the first string: ");
   scanf("%s", s1);
  // printf("Enter the second string: ");
   scanf("%s", s2);

   int m = strlen(s1);
   int n = strlen(s2);

   // Initialize the DP table
   for (int i = 0; i <= m; i++) {
      for (int j = 0; j <= n; j++) {
         if (i == 0 || j == 0) {
            dp[i][j] = 0; // Base case: LCS length is 0 if either string is empty
```

```
            } else if (s1[i - 1] == s2[j - 1]) {
                dp[i][j] = dp[i - 1][j - 1] + 1; // Characters match
            } else {
                dp[i][j] = (dp[i - 1][j] > dp[i][j - 1]) ? dp[i - 1][j] : dp[i][j - 1]; // Characters don't match
            }
        }
    }

    // Print the length of LCS
    printf("%d\n", dp[m][n]);

    return 0;
}
```

INPUT:

s1: ggtabe

s2: tgatasb

OUTPUT:

	Input	Expected	Got	
✔	aab azb	2	2	✔
✔	ABCD ABCD	4	4	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

4-DP-Longest non-decreasing Subsequence

AIM:

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

INPUT:

Input:9

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

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

CODE:

```
#include <stdio.h>


int main() {
   int n;


   // Input the size of the sequence
   //printf("Enter the size of the sequence: ");
   scanf("%d", &n);


   int sequence[n];
   //printf("Enter the sequence: ");
   for (int i = 0; i < n; i++) {
      scanf("%d", &sequence[i]);
   }


   // DP array to store the length of LNDS ending at each position
   int dp[n];
```

```
// Initialize all lengths to 1 (each element is a subsequence of length 1)
for (int i = 0; i < n; i++) {
   dp[i] = 1;
}

// Compute LNDS using dynamic programming
for (int i = 1; i < n; i++) {
   for (int j = 0; j < i; j++) {
      if (sequence[i] >= sequence[j] && dp[i] < dp[j] + 1) {
         dp[i] = dp[j] + 1;
      }
   }
}

// Find the maximum value in the dp array
int maxLength = 0;
for (int i = 0; i < n; i++) {
   if (dp[i] > maxLength) {
      maxLength = dp[i];
   }
}

// Print the length of the Longest Non-Decreasing Subsequence
printf("%d\n", maxLength);
```

```
    return 0;
}
```

OUTPUT:

	Input	Expected	Got	
✔	9 -1 3 4 5 2 2 2 2 3	6	6	✔
✔	7 1 2 2 4 5 7 6	6	6	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.