REG NO: 230701504  
NAME : KAAVIYA.R

DEPT : CSE – A SECTION

**DYNAMIC PROGRAMMING**

**QUESTION4.A AIM :**



**ALGORITHM :**

Step 1: Start

Step 2: Input an integer n

Step 3: Initialize an array dp of size n+1

Step 4: Set dp[0] to 1

Step 5: For each index i from 1 to n, set dp[i] to 0

Step 6: For each index i from 1 to n, do Steps 7 and 8

Step 7: Add dp[i - 1] to dp[i]

Step 8: If i >= 3, add dp[i - 3] to dp[i]

Step 9: Print dp[n] Step 10: Stop



**OUTPUT :**



**RESULT:**

The above program is executed successfully.

**QUESTION 4.B AIM:**



**ALGORITHM :**

Step 1: Start

Step 2: Input an integer n

Step 3: Initialize a 2D array board of size n x n

Step 4: For each row i from 0 to n-1, and each column j from 0 to n-1, input board[i][j]

Step 5: Call maxMonetaryPath(n, board) and store the result in result

Step 6: Print result Step 7: Stop

**PROGRAM:**



**OUTPUT:**



**RESULT :**

The above program is executed successfully.

**QUESTION 4.C** **AIM**



**ALGORITHM :**

Step 1: Start

Step 2: Input two strings s1 and s2

Step 3: Calculate the lengths len1 of s1 and len2 of s2

Step 4: Initialize a 2D array dp of size (len1 + 1) x (len2 + 1)

Step 5: For each index i from 0 to len1, and each index j from 0 to len2, do Steps 6-8

Step 6: If i == 0 or j == 0, set dp[i][j] = 0

Step 7: If s1[i-1] == s2[j-1], set dp[i][j] = dp[i-1][j-1] + 1

Step 8: Otherwise, set dp[i][j] to the maximum of dp[i][j-1] and dp[i-1][j]

Step 9: Print dp[len1][len2]

Step 10: Stop

**PROGRAM:**



**OUTPUT :**



**RESULT:**

The above program is executed successfully.

**QUESTION 4.D AIM :**



**ALGORITHM :**

Step 1: Start

Step 2: Input an integer n

Step 3: Initialize an array arr of size n

Step 4: For each index i from 0 to n-1, input arr[i]

Step 5: Call subsequence(arr, n) and store the result in result

Step 6: Print result Step 7: Stop

PROGRAM:



**OUTPUT :**



**RESULT :**

The above program is executed successfully.