Ex. No: 5 **Date: 10.09.24**

Register No.: 230701386 **Name: R YASHVINTHINI**

Dynamic Programming

5.a. Playing with Numbers

Aim: 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

Algorithm:

1) Initialize an array ways of size n+1 to store the number of ways to represent each number from 0 to n.

2) Set ways[0] to 1

3) Iterate from 1 to n and for each number i, calculate the number of ways to represent i using the numbers 1 and 3.

4) For each i, add the number of ways to represent i-1 and i-3 to ways[i].

5) Return ways[n] as the result.

Program:

```
#include <stdio.h>
int countWays(int n) {
    int ways[n + 1];
   ways[0] = 1;  // Base case: 1 way to represent 0

   for (int i = 1; i <= n; i++) {
      ways[i] = 0;
      if (i >= 1) ways[i] += ways[i - 1];
      if (i >= 3) ways[i] += ways[i - 3];
   }
   return ways[n];
}

int main() {
   int n;
   printf("Enter a positive integer: ");
   scanf("%d", &n);
    int result = countWays(n);
   printf("%d", result);
}
```

Output:

	Input	Expected	Got	
✔	6	6	6	✔
✔	25	8641	8641	✔
✔	100	243828195967216 29	243828195967216 29	✔

5.b. Playing with chessboard

Aim: Ram is given with an n*n chessboard with each cell with a monetary value. Ram stands at the (0,0), that the position of the top left white rook. He is been 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*n chessboard values

Output Format

Print Maximum monetary value of the path

Algorithm:

1) Initialize a 2D array dp of size n*n to store the maximum monetary value that can be collected up to each cell.

2) Set dp[0][0] to the monetary value of the starting cell (0,0).

3) Iterate through each cell (i, j) in the chessboard:

- If i > 0, update dp[i][j] to be the maximum of dp[i][j] and dp[i-1][j] + value[i][j].
- If j > 0, update dp[i][j] to be the maximum of dp[i][j] and dp[i][j-1] + value[i][j].

5) The value at dp[n-1][n-1] will be the maximum monetary value that can be collected.

Program:

```
#include <stdio.h>
#define MAX 100
int max(int a, int b) {
   return (a > b) ? a : b;
}
int maxMonetaryPath(int n, int value[MAX][MAX]) {
   int dp[MAX][MAX];
   dp[0][0] = value[0][0];

   for (int i = 0; i < n; i++) {
      for (int j = 0; j < n; j++) {
         if (i > 0) {
            dp[i][j] = max(dp[i][j], dp[i-1][j] + value[i][j]);
         }
         if (j > 0) {
            dp[i][j] = max(dp[i][j], dp[i][j-1] + value[i][j]);
         }
      }
   }
  return dp[n-1][n-1];
}
int main() {
   int n;
   int value[MAX][MAX];
   scanf("%d", &n);
   for (int i = 0; i < n; i++)
```

```
{
    for (int j = 0; j < n; j++)
    {
        scanf("%d", &value[i][j]);
     }
}
int result = maxMonetaryPath(n, value);
printf("%d", result);
return 0;
```

}

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	✔