Dynamic Programming

1.

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

Algorithm:

1. Start
2. Input integer n
3. Define array dp[n+1]
4. Set dp[0] = 1 and initialize dp[i] = 0 for i = 1 to n
5. For i = 1 to n:
6. Update dp[i] += dp[i - 1]
7. If i >= 3, update dp[i] += dp[i - 3]
8. Output dp[n]
9. End

Program:

```
#include<stdio.h>
int main() {
   int n;
   scanf("%d", &n);
   long dp[n+1];
   dp[0] = 1;
   for (int i = 1; i <= n; i++) {
      dp[i] = 0;
   }
   for (int i = 1; i <= n; i++) {
      dp[i] += dp[i - 1];
```

```
        if (i >= 3) {
            dp[i] += dp[i - 3];
        }
    }
    printf("%ld\n", dp[n]);
    return 0;
}
```

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

Passed all tests! ✔

2.

Playing with Chessboard:

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. Start
2. Input integer n
3. Input board[n][n]
4. For j = 1 to n-1, update dp[0][j] = dp[0][j-1] + board[0][j]
5. For i = 1 to n-1, update dp[i][0] = dp[i-1][0] + board[i][0]
6. For i = 1 to n-1 and j = 1 to n-1:
7. Update dp[i][j] = board[i][j] + max(dp[i-1][j], dp[i][j-1])
8. Output dp[n-1][n-1]
9. End

Program:

```
#include<stdio.h>
int max(int a,int b) {
   return(a>b) ? a:b;
}

int maxMonetaryPath(int n,int board[n][n]){
   int dp[n][n];
   dp[0][0]=board[0][0];

   for(int j=1;j<n;j++){
      dp[0][j]=dp[0][j-1]+board[0][j];
   }
```

```
    for (int i=1;i<n;i++) {
        dp[i][0]=dp[i-1][0]+board[i][0];
    }

    for (int i=1;i<n;i++) {
        for (int j=1;j<n;j++) {
            dp[i][j]=board[i][j]+max(dp[i-1][j],dp[i][j-1]);
        }
    }
    return dp[n-1][n-1];
}

int main(){
    int n;
    scanf("%d",&n);
    int board[n][n];
    for (int i=0;i<n;i++){
        for (int j=0;j<n;j++){
            scanf("%d",&board[i][j]);
        }
    }

    int result=maxMonetaryPath(n,board);
    printf("%d\n",result);
}
```

	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! ✔

3.

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	g	t	a	b	
s2	g	x	t	x	a	y	b

The length is 4

Solveing it using Dynamic Programming

For example:

Input	Result
aab azb	2

Algorithm:

1. Start
2. Input strings s1 and s2
3. Define dp[len1 + 1][len2 + 1] for dynamic programming
4. Calculate lengths: len1 = strlen(s1) and len2 = strlen(s2)
5. For i = 0 to len1:
6. For j = 0 to len2:
7. If i == 0 or j == 0, set dp[i][j] = 0
8. If s1[i-1] == s2[j-1], set dp[i][j] = dp[i-1][j-1] + 1
9. Else, set dp[i][j] = max(dp[i-1][j], dp[i][j-1])
10. Output dp[len1][len2]
11. End

Program:

```
#include<stdio.h>
#include<string.h>
int main()
{
   char s1[10],s2[10];
   scanf("%s",s1);
```

```
    scanf("%s",s2);
    int len1=strlen(s1);
    int len2=strlen(s2);
    int dp[len1 + 1][len2 + 1];
    for(int i=0;i<=len1;i++)
    {
       for(int j=0;j<=len2;j++)
       {
          if(i==0||j==0)
          {
             dp[i][j]=0;
          }
          else if(s1[i-1]==s2[j-1])
          {
             dp[i][j]=dp[i-1][j-1]+1;
          }
          else{
             if(dp[i][j-1]>dp[i-1][j])
                dp[i][j]=dp[i][j-1];
             else
                dp[i][j]=dp[i-1][j];
             }
       }
    }
    printf("%d",dp[len1][len2]);
}
```

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

Passed all tests! ✔

4.

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

Algorithm:

1. Start
2. Input integer n
3. Input array arr[n]
4. Define array dp[n] for storing subsequence lengths
5. Set all values in dp to 1 (each element is a subsequence of length 1)
6. Set maxlen = 1
7. Update maxlen = max(maxlen, dp[i])
8. Output maxlen
9. End

Program:

```
#include <stdio.h>
int subsequence(int arr[],int n){
   int dp[n];
   int maxlen=1;
   for (int i=0;i<n;i++){
      dp[i]=1;
   }
   for (int i=1;i<n;i++){
      for(int j=0;j<i;j++){
         if(arr[i]>=arr[j] && dp[i]<dp[j]+1){
            dp[i]=dp[j]+1;
         }
      }
```

```
        if(maxlen<dp[i]){
            maxlen=dp[i];
        }
    }
    return maxlen;
}

int main(){
    int n;
    scanf("%d",&n);
    int arr[n];
    for (int i=0;i<n;i++){
        scanf("%d",&arr[i]);
    }
    int result=subsequence(arr,n);
    printf("%d",result);
}
```

	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! ✔