## 1-DP-Playing with Numbers

| Started on | Friday, 31 October 2025, 1:29 AM |
| State | Finished |
| Completed on | Friday, 31 October 2025, 1:30 AM |
| Time taken | 13 secs |
| Grade | **10.00** out of 10.00 (**100%**) |

**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

Sample Output

6

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

```c
#include <stdio.h>

unsigned long long countWays(int n) {
    unsigned long long dp[n + 1];
    dp[0] = 1;

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

    return dp[n];
}

int main() {
    int n;
    scanf("%d", &n);
    printf("%llu\n", countWays(n));
    return 0;
}
```

| | 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

**Question 1**  Correct  Mark 10.00 out of 10.00  ⚑ Flag question

**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

```c
#include <stdio.h>

#define MAX 100

int main() {
    int n;
    scanf("%d", &n);

    int board[MAX][MAX];
    int dp[MAX][MAX];

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

    dp[0][0] = board[0][0];

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (i == 0 && j == 0) continue;
            int from_top = (i > 0) ? dp[i - 1][j] : 0;
            int from_left = (j > 0) ? dp[i][j - 1] : 0;
            dp[i][j] = (from_top > from_left ? from_top : from_left) + board[i][j];
        }
    }

    printf("%d\n", dp[n - 1][n - 1]);
    return 0;
}
```

| | 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 | 28 | 28 | ✔ |

```
26      }
27    }
28
29    printf("%d\n", dp[n - 1][n - 1]);
30    return 0;
31 }
32
```

| | 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.

# ☑ 3-DP-Longest Common Subsequence

| | |
|---|---|
| **Started on** | Tuesday, 21 October 2025, 3:49 PM |
| **State** | Finished |
| **Completed on** | Tuesday, 21 October 2025, 3:50 PM |
| **Time taken** | 46 secs |
| **Marks** | 1.00/1.00 |
| **Grade** | **10.00** out of 10.00 **(100%)** |

**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 | **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<br>azb | 2 |

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

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

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

#define MAX 1000

int max(int a, int b) {
    return (a > b) ? a : b;
}

int main() {
    char s1[MAX], s2[MAX];
    scanf("%s %s", s1, s2);

    int len1 = strlen(s1);
    int len2 = strlen(s2);

    int dp[MAX][MAX];

    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
                dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
        }
    }

    printf("%d\n", dp[len1][len2]);
    return 0;
}
```

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

Passed all tests! ✔

**Correct**

Marks for this submission: 1.00/1.00.

# ☑ 4-DP-Longest non-decreasing Subsequence

| | |
|---|---|
| **Started on** | Tuesday, 21 October 2025, 3:52 PM |
| **State** | Finished |
| **Completed on** | Tuesday, 21 October 2025, 3:52 PM |
| **Time taken** | 38 secs |
| **Marks** | 1.00/1.00 |
| **Grade** | **10.00** out of 10.00 (**100%**) |

**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 %)

```c
1   #include <stdio.h>
2
3   #define MAX 1000
4
5   int max(int a, int b) {
6       return (a > b) ? a : b;
7   }
8
9   int main() {
10      int n;
11      scanf("%d", &n);
12
13      int arr[MAX];
14      for (int i = 0; i < n; i++) {
15          scanf("%d", &arr[i]);
16      }
```

```c
 5  int max(int a, int b) {
 6      return (a > b) ? a : b;
 7  }
 8
 9  int main() {
10      int n;
11      scanf("%d", &n);
12
13      int arr[MAX];
14      for (int i = 0; i < n; i++) {
15          scanf("%d", &arr[i]);
16      }
17
18      int dp[MAX];
19      for (int i = 0; i < n; i++) {
20          dp[i] = 1;
21      }
22
23      for (int i = 1; i < n; i++) {
24          for (int j = 0; j < i; j++) {
25              if (arr[i] >= arr[j]) {
26                  dp[i] = max(dp[i], dp[j] + 1);
27              }
28          }
29      }
30
31      int result = 0;
32      for (int i = 0; i < n; i++) {
33          result = max(result, dp[i]);
34      }
35
36      printf("%d\n", result);
37      return 0;
38  }
39
```

| | 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! ✔

Marks for this submission: 1.00/1.00.