

Dynamic Programming

1-DP-Playing with Numbers

Playing with Numbers:

Ram and Sita are playing with numbers by giving puzzles to each other. Now it was Ram's turn, so he gave Sita a positive integer 'n' and two numbers 1 and 3. 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 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

Answer: (penalty regime: 0 %)

```
1 #include <stdio.h>
2
3 v int main() {
4     int n;
5     scanf("%d", &n);
6
7 v     if (n < 0) {
8         printf("0\n");
9         return 0;
10    }
11 v    if (n == 0) {
12        printf("1\n");
13        return 0;
14    }
15
16    long long a = 1;
17    long long b = 1;
18    long long c = 1;
19    long long curr = 0;
20
21 v    for (int i = 3; i <= n; i++) {
22        curr = c + a;
23        a = b;
24        b = c;
25        c = curr;
26    }
27    if (n == 1)
28        printf("%lld\n", b);
29    else if (n == 2)
30        printf("%lld\n", c);
31    else
32        printf("%lld\n", curr);
33
34    return 0;
35 }
36 }
```

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

Passed all tests! ✓

Correct

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 $n \times n$ chessboard with each cell with a monetary value. Ram stands at the $(0,0)$, that is the position of the top left white rook. He is given a task $(n-1, n-1)$ 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 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 $n \times n$ chessboard values

Output Format

Print Maximum monetary value of the path

Answer: (penalty regime: 0 %)

```
1 #include <stdio.h>
2
3 int main() {
4     int n;
5     scanf("%d", &n);
6
7     int arr[n][n];
8     for (int i = 0; i < n; i++)
9         for (int j = 0; j < n; j++)
10            scanf("%d", &arr[i][j]);
11     int dp[n][n];
12     dp[0][0] = arr[0][0];
13     for (int j = 1; j < n; j++)
14         dp[0][j] = dp[0][j - 1] + arr[0][j];
15     for (int i = 1; i < n; i++)
16         dp[i][0] = dp[i - 1][0] + arr[i][0];
17     for (int i = 1; i < n; i++) {
18         for (int j = 1; j < n; j++) {
19             int maxPrev = dp[i - 1][j] > dp[i][j - 1] ? dp[i - 1][j] : dp[i][j - 1];
20             dp[i][j] = arr[i][j] + maxPrev;
21         }
22     }
23     printf("%d", dp[n - 1][n - 1]);
24
25     return 0;
26 }
27 }
```

	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

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

Solving it using Dynamic Programming

For example:

Input	Result
aab	2
azb	

Answer: (penalty regime: 0 %)

```
1 #include <stdio.h>
2 #include <string.h>
3
4 int max(int a, int b) {
5     return (a > b) ? a : b;
6 }
7
8 int main() {
9     char s1[100], s2[100];
10    scanf("%s %s", s1, s2);
11
12    int n = strlen(s1);
13    int m = strlen(s2);
14    int dp[n + 1][m + 1];
15    for (int i = 0; i <= n; i++) {
16        for (int j = 0; j <= m; j++) {
17            dp[i][j] = 0;
18        }
19    }
20    for (int i = 1; i <= n; i++) {
21        for (int j = 1; j <= m; j++) {
22            if (s1[i - 1] == s2[j - 1])
23                dp[i][j] = 1 + dp[i - 1][j - 1];
24            else
25                dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
26        }
27    }
28    printf("%d", dp[n][m]);
29
30    return 0;
31 }
```

	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

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 %)

```
1 #include <stdio.h>
2 int max(int a, int b) {
3     return (a > b) ? a : b;
4 }
5 int main() {
6     int n;
7     scanf("%d", &n);
8     int arr[n];
9     for (int i = 0; i < n; i++)
10        scanf("%d", &arr[i]);
11     int dp[n];
12     for (int i = 0; i < n; i++)
13        dp[i] = 1;
14     for (int i = 1; i < n; i++) {
15         for (int j = 0; j < i; j++) {
16             if (arr[i] >= arr[j]) {
17                 dp[i] = max(dp[i], dp[j] + 1);
18             }
19         }
20     }
21     int maxLength = 0;
22     for (int i = 0; i < n; i++)
23         if (dp[i] > maxLength)
24             maxLength = dp[i];
25     printf("%d", maxLength);
26     return 0;
27 }
28 }
```

	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! ✓