	Input	Expected	Got
✔	5 67 34 12 98 78	12 34 67 78 98	12 34 67 78 98
✔	10 1 56 78 90 32 56 11 10 90 114	1 10 11 32 56 56 78 90 90 114	1 10 11 32 56 56 78 90 90
✔	12 9 8 7 6 5 4 3 2 1 10 11 90	1 2 3 4 5 6 7 8 9 10 11 90	1 2 3 4 5 6 7 8 9 10 11 9

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 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

```
#include <stdio.h>


long long int countWays(int n) {
```

```
    long long int a[n + 1];

    a[0] = 1;
    a[1] = 1;


    if (n >= 2) {
        a[2] = 1;
    }

    if (n >= 3) {
        a[3] = 2;
    }

    for (int i = 4; i <= n; i++) {
        a[i] = a[i - 1] + a[i - 3];
    }

    return a[n];
}

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

    long long int result = countWays(n);
    printf("%lld",result);

    return 0;
}
```

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

Passed all tests! ✔

2-DP-Playing with chessboard

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

```
#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-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

Solveing it using Dynamic Programming

For example:

Input	Result
aab azb	2

program:

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

int longestCommonSubsequence(char s1[], char s2[]) {
    int m = strlen(s1);
    int n = strlen(s2);

    int dp[m + 1][n + 1];

    // Initialize the DP table with base cases
    for (int i = 0; i <= m; i++) {
```

```
        for (int j = 0; j <= n; 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] = (dp[i - 1][j] > dp[i][j - 1]) ? dp[i - 1][j] : dp[i][j - 1];
            }
        }
    }

    return dp[m][n];
}

int main() {
    char s1[100], s2[100];

    scanf("%s", s1);

    scanf("%s", s2);

    int result = longestCommonSubsequence(s1, s2);
    printf("%d", result);

    return 0;
}
```

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

Passed all tests! ✔

4-DP-Longest non-decreasing Subsequence

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

program:

```
#include <stdio.h>


int longestNonDecreasingSubsequence(int n, int sequence[]) {
    int dp[n];
    int maxLength = 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 (sequence[j] <= sequence[i]) {
                dp[i] = (dp[i] > dp[j] + 1) ? dp[i] : dp[j] + 1;
            }
        }

        maxLength = (maxLength > dp[i]) ? maxLength : dp[i];
    }

    return maxLength;
}

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

    int sequence[n];

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

    int result = longestNonDecreasingSubsequence(n, sequence);
    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! ✔

1-Finding Duplicates-O(n^2) Time Complexity,O(1) Space Complexity

Find Duplicate in Array.

Given a read only array of n integers between 1 and n, find one number that repeats.

Input Format:

First Line - Number of elements

n Lines - n Elements

Output Format:

Element x - That is repeated

For example:

Input	Result
5 1 1 2 3 4	1

program:

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