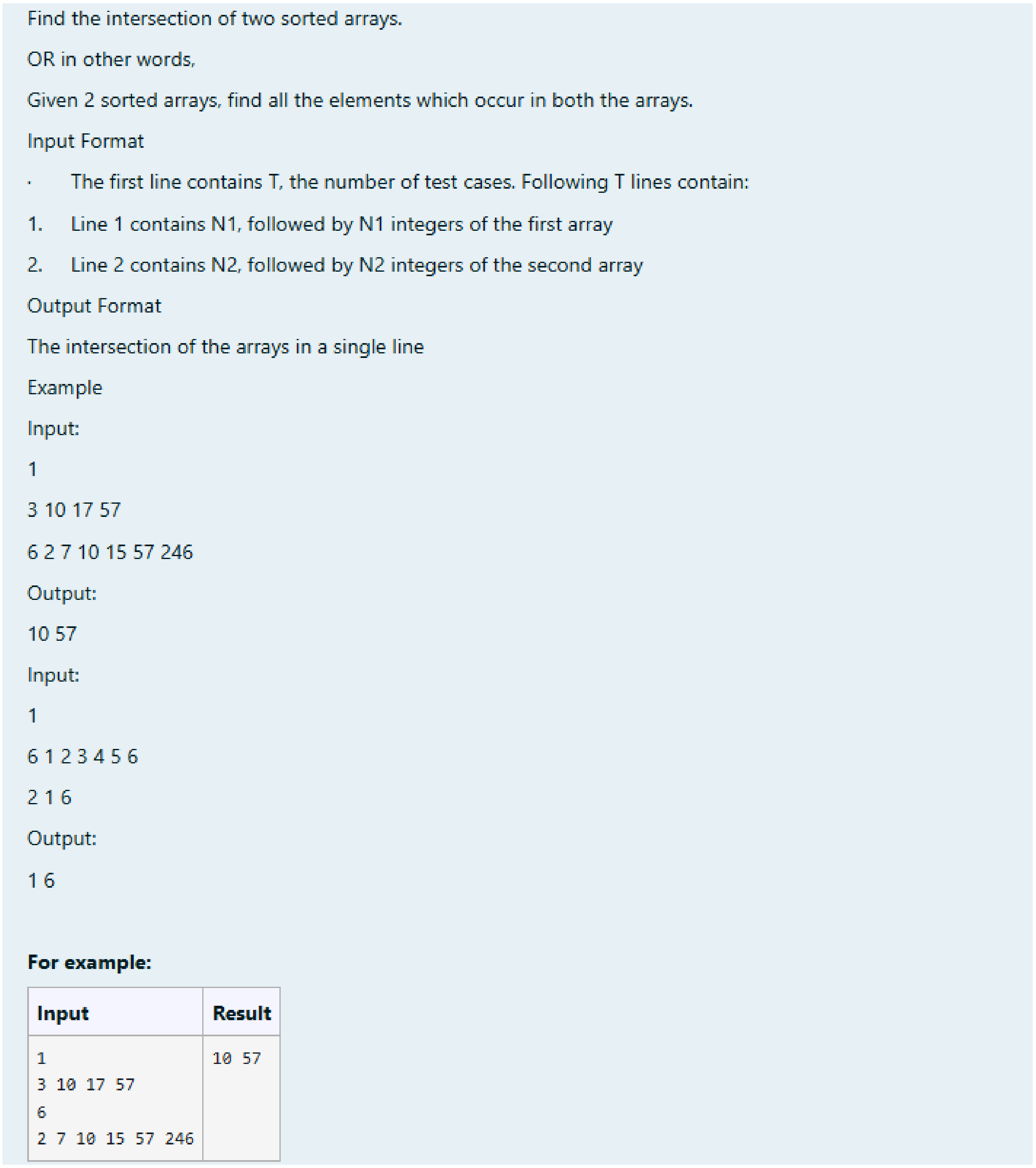
Competitive Programming (cont.)

3)



Find the intersection of two sorted arrays.

OR in other words,

Given 2 sorted arrays, find all the elements which occur in both the arrays.

Input Format

- The first line contains T, the number of test cases. Following T lines contain:

1. Line 1 contains N1, followed by N1 integers of the first array
2. Line 2 contains N2, followed by N2 integers of the second array

Output Format

The intersection of the arrays in a single line

Example

Input:

1

3 10 17 57

6 2 7 10 15 57 246

Output:

10 57

Input:

1

6 1 2 3 4 5 6

2 1 6

Output:

1 6

For example:

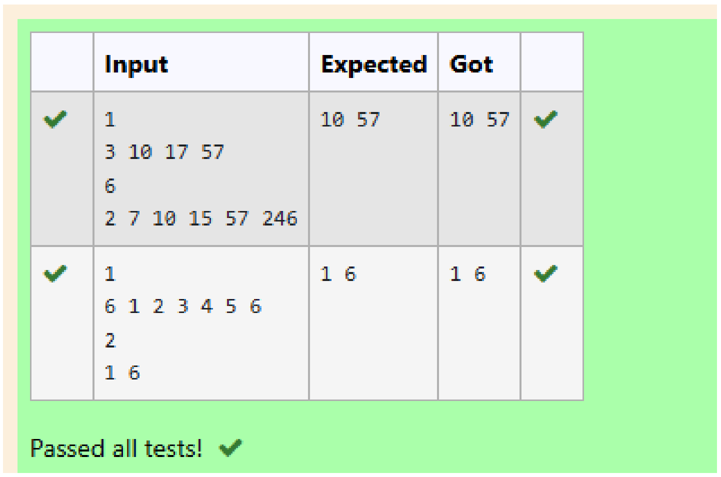
Input	Result

`1`
`3 10 17 57`
`6`
`2 7 10 15 57 246`
	`10 57`

Code:

```
#include <stdio.h>
void findIntersection(int arr1[], int n1, int arr2[], int n2) {
    int i = 0, j = 0;
    int first = 1;
    while (i < n1 && j < n2) {
        if (arr1[i] < arr2[j]) {
            i++;
        } else if (arr1[i] > arr2[j]) {
            j++;
        } else {
            if (first) {
                printf("%d", arr1[i]);
                first = 0;
            } else {
                printf(" %d", arr1[i]);
            }
            i++;
            j++;
        }
    }
    if (first) {
        printf(" ");
    }
}
int main() {
    int T;
    scanf("%d", &T);
    while (T--) {
        int n1;
        scanf("%d", &n1);
        int arr1[n1];
        for (int i = 0; i < n1; i++) {
            scanf("%d", &arr1[i]);
        }
        int n2;
        scanf("%d", &n2);
        int arr2[n2];
        for (int i = 0; i < n2; i++) {
            scanf("%d", &arr2[i]);
        }
        findIntersection(arr1, n1, arr2, n2);
        printf("\n");
    }
    return 0;
}
```

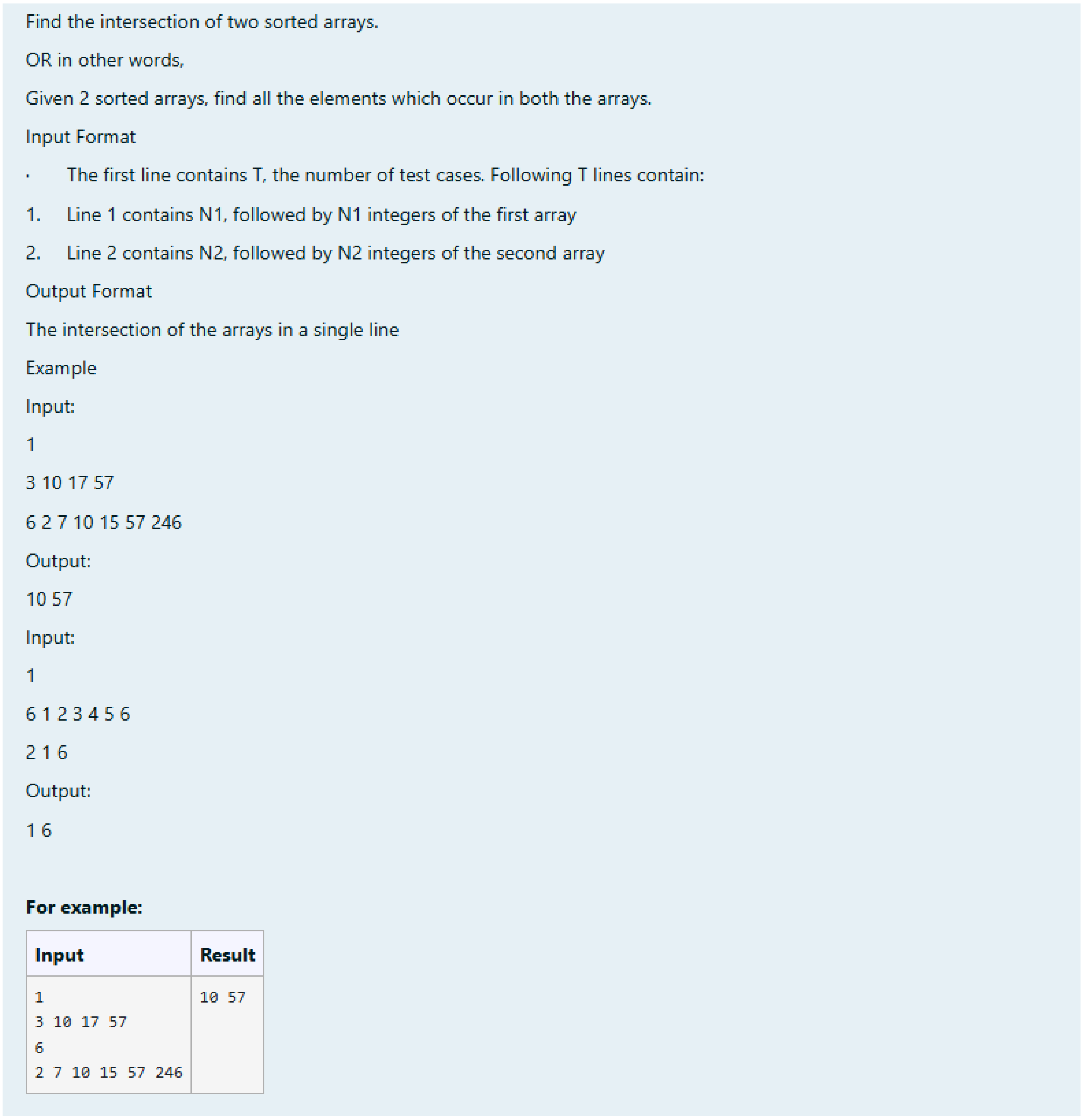
OUTPUT:



	Input	Expected	Got	
✓	1 3 10 17 57 6 2 7 10 15 57 246	10 57	10 57	✓
✓	1 6 1 2 3 4 5 6 2 1 6	1 6	1 6	✓

Passed all tests! ✓

4)



Find the intersection of two sorted arrays.

OR in other words,

Given 2 sorted arrays, find all the elements which occur in both the arrays.

Input Format

- The first line contains T, the number of test cases. Following T lines contain:

1. Line 1 contains N1, followed by N1 integers of the first array
2. Line 2 contains N2, followed by N2 integers of the second array

Output Format

The intersection of the arrays in a single line

Example

Input:

1

3 10 17 57

6 2 7 10 15 57 246

Output:

10 57

Input:

1

6 1 2 3 4 5 6

2 1 6

Output:

1 6

For example:

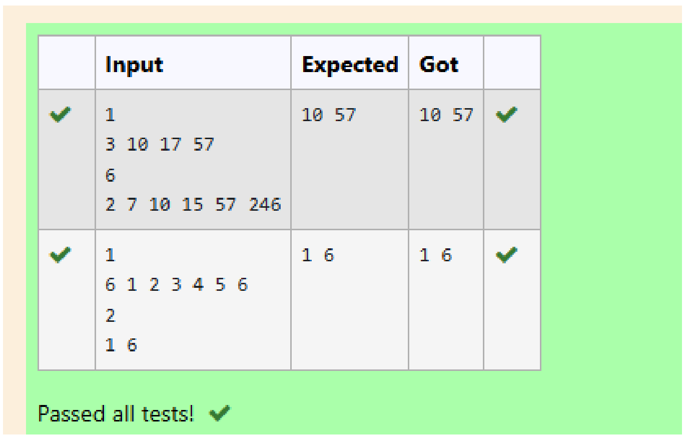
Input	Result

`1`
`3 10 17 57`
`6`
`2 7 10 15 57 246`
	`10 57`

Code:

```
#include <stdio.h>
void findIntersection(int arr1[], int n1, int arr2[], int n2) {
    int i = 0, j = 0;
    int first = 1;
    while (i < n1 && j < n2) {
        if (arr1[i] < arr2[j]) {
            i++;
        } else if (arr1[i] > arr2[j]) {
            j++;
        } else {
            if (first) {
                printf("%d", arr1[i]);
                first = 0;
            } else {
                printf(" %d", arr1[i]);}
            i++;
            j++;
        }
    }
}
int main() {
    int T;
    scanf("%d", &T);

    while (T--) {
        int n1;
        scanf("%d", &n1);
        int arr1[n1];

        for (int i = 0; i < n1; i++) {
            scanf("%d", &arr1[i]);
        }
        int n2;
        scanf("%d", &n2);
        int arr2[n2];

        for (int i = 0; i < n2; i++) {
            scanf("%d", &arr2[i]);
        }
        findIntersection(arr1, n1, arr2, n2);
        printf("\n");
    }
    return 0;
}
```

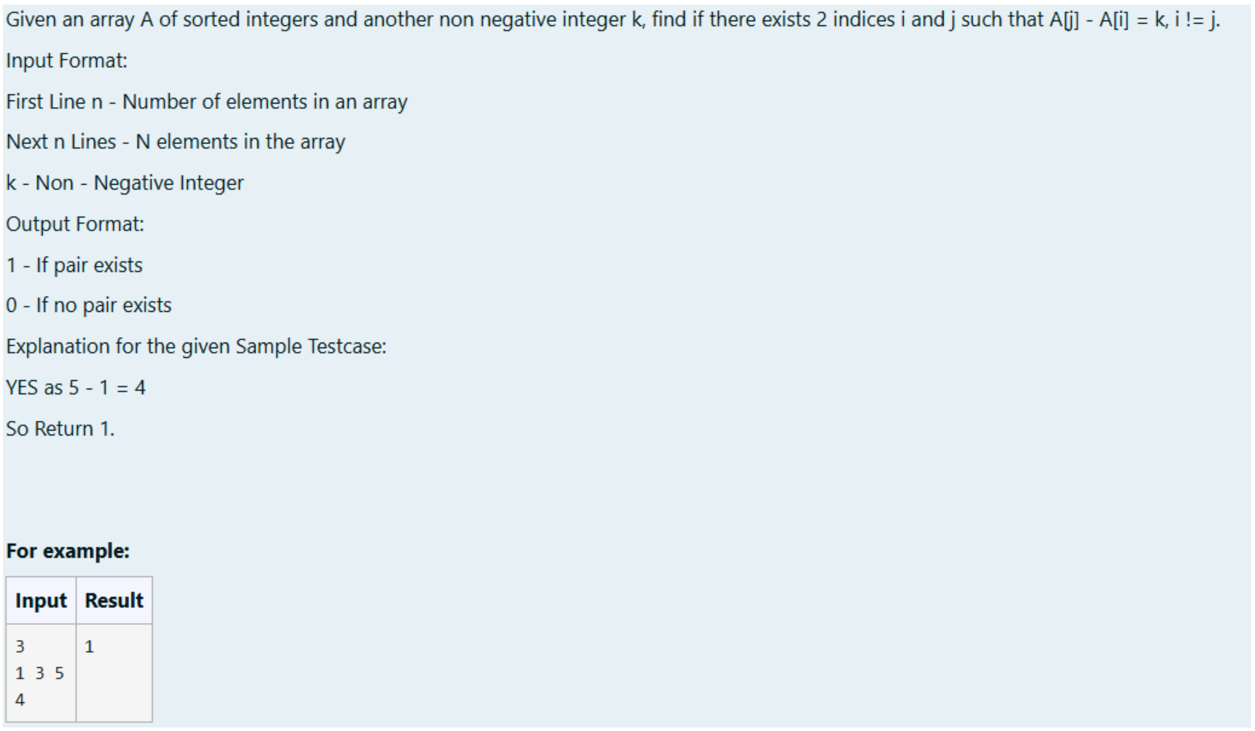
Output:



	Input	Expected	Got	
✓	1 3 10 17 57 6 2 7 10 15 57 246	10 57	10 57	✓
✓	1 6 1 2 3 4 5 6 2 1 6	1 6	1 6	✓

Passed all tests! ✓

5)



Given an array A of sorted integers and another non negative integer k, find if there exists 2 indices i and j such that A[j] - A[i] = k, i != j.

Input Format:

First Line n - Number of elements in an array

Next n Lines - N elements in the array

k - Non - Negative Integer

Output Format:

1 - If pair exists

0 - If no pair exists

Explanation for the given Sample Testcase:

YES as 5 - 1 = 4

So Return 1.

For example:

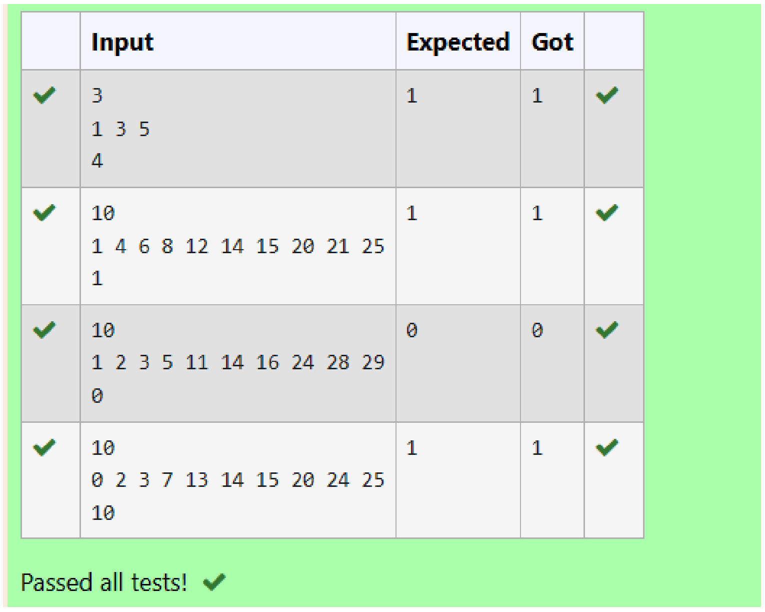
Input	Result

3
1 3 5
4
	1

Code:

```
#include<stdio.h>
#include<stdlib.h>
void array(int n,int a[],int k){
   for(int i=0;i<n;i++){
      for(int j=i+1;j<n;j++){
           if(abs(a[i]-a[j])==k){
               printf("1");
               return;
           }
      }
   }
   printf("0");
}
int main(){
   int n,k;
   scanf("%d",&n);
   int a[n];
   for(int i=0;i<n;i++){
      scanf("%d",&a[i]);
   }
   scanf("%d",&k);
   array(n,a,k);
   return 0;
}
```

Output:

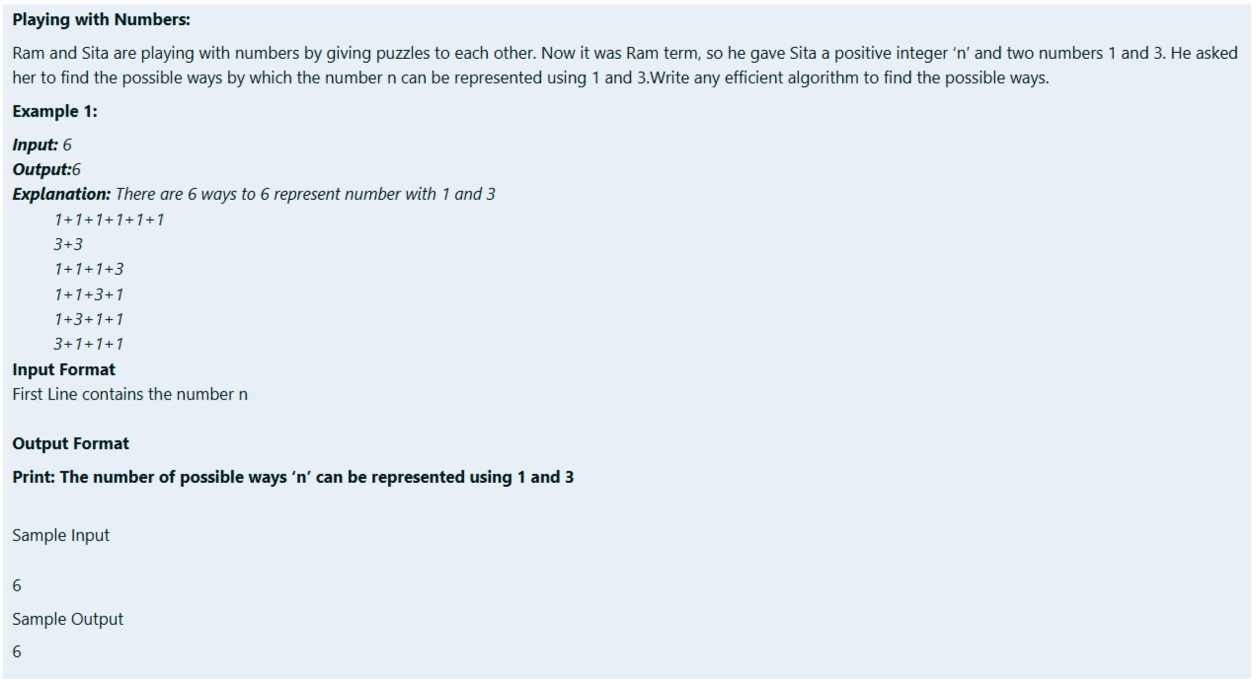


	Input	Expected	Got	
✓	3 1 3 5 4	1	1	✓
✓	10 1 4 6 8 12 14 15 20 21 25 1	1	1	✓
✓	10 1 2 3 5 11 14 16 24 28 29 0	0	0	✓
✓	10 0 2 3 7 13 14 15 20 24 25 10	1	1	✓

Passed all tests! ✓

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

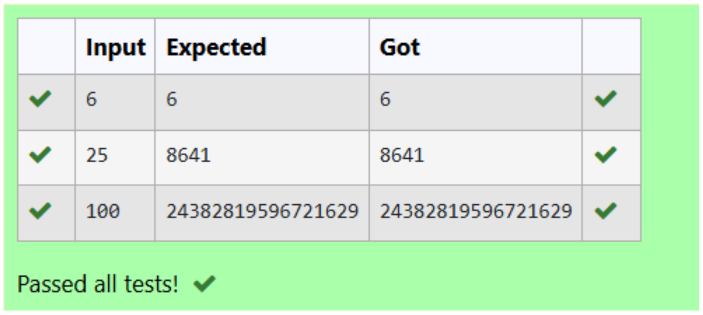
Sample Output

6

Code:

```
#include <stdio.h>

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

    if (n < 0) {
        printf("Invalid input\n");
        return 1;
    }

    long long dp[n + 1];

    dp[0] = 1;
    dp[1] = 1;
    dp[2] = 1;
    if (n >= 3) dp[3] = 2;
    for (int i = 4; i <= n; i++) {
        dp[i] = dp[i - 1] + dp[i - 3];
    }

    printf("%lld\n", dp[n]);

    return 0;
}
```

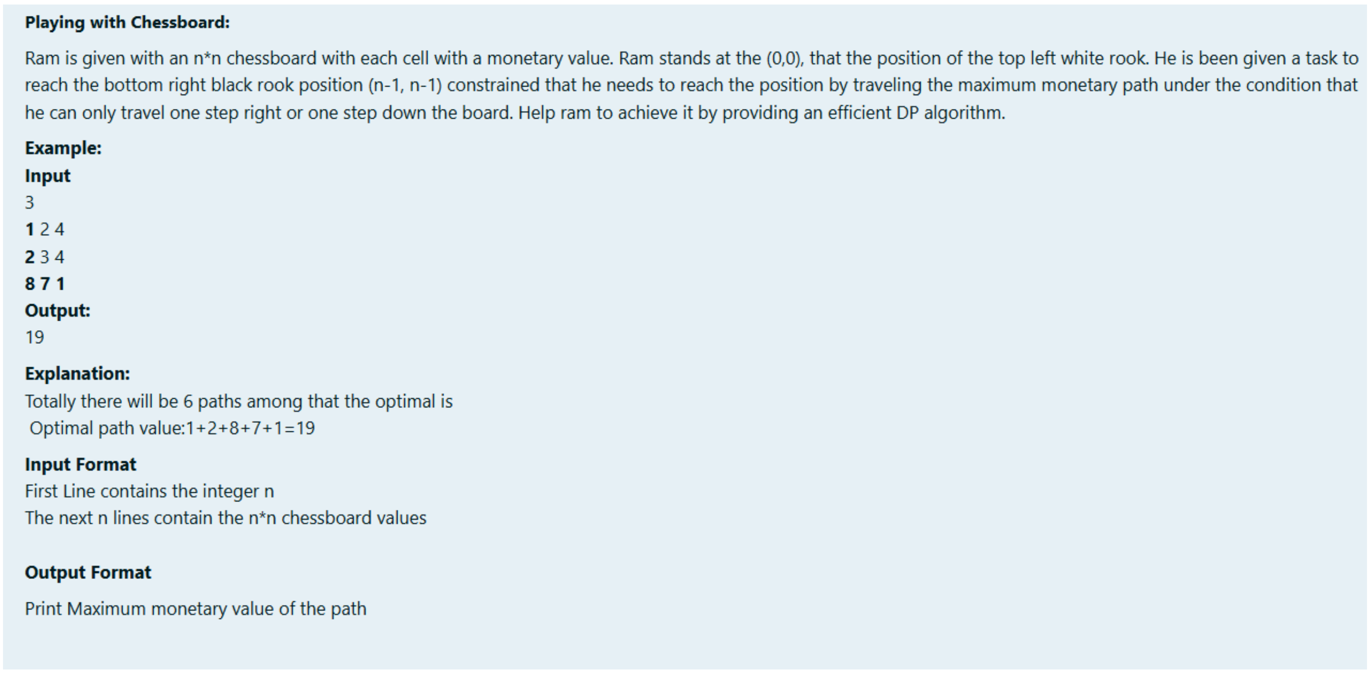
Output:



	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

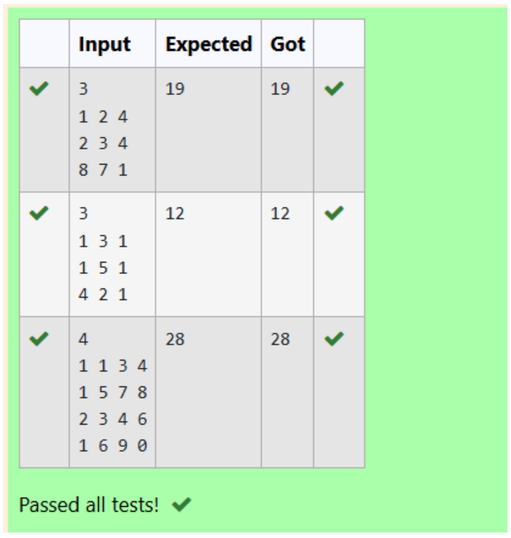
Output Format

Print Maximum monetary value of the path

Code:

```
#include <stdio.h>
#define MAX 100
int main() {
    int n;
    int board[MAX][MAX];
    int dp[MAX][MAX];
    scanf("%d", &n);
    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 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] + (dp[i-1][j] > dp[i][j-1] ? dp[i-1][j] : dp[i][j-1]);}}
    printf("%d\n", dp[n-1][n-1]);
    return 0;
}
```

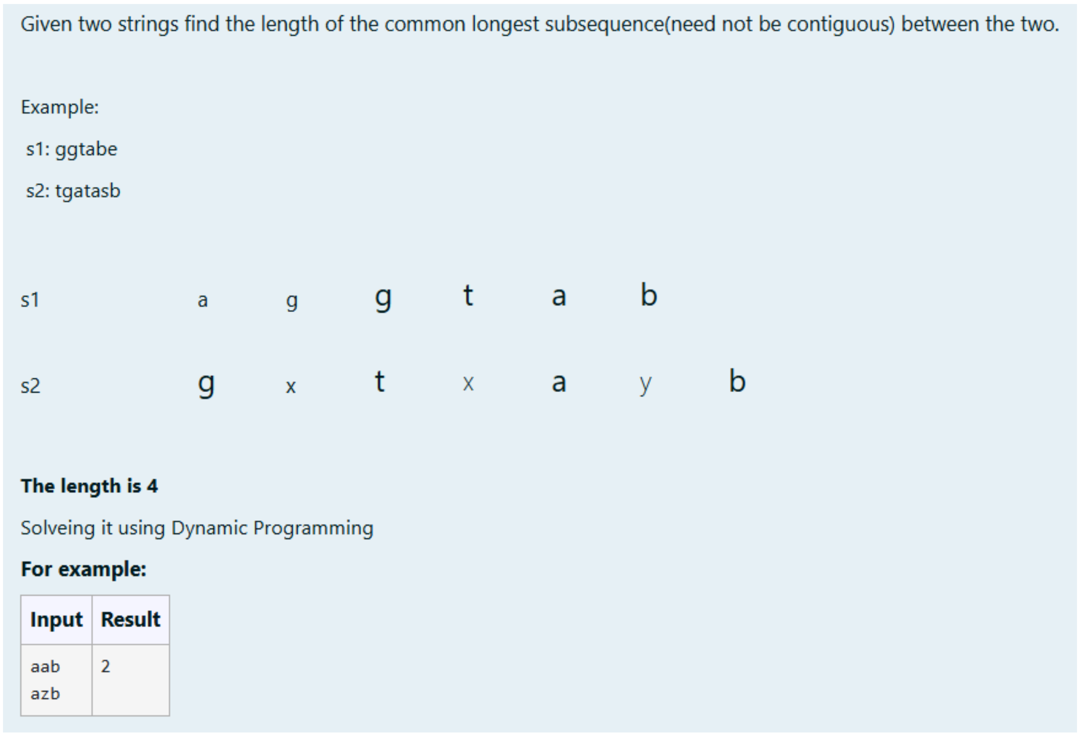
Output:



	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	✓

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

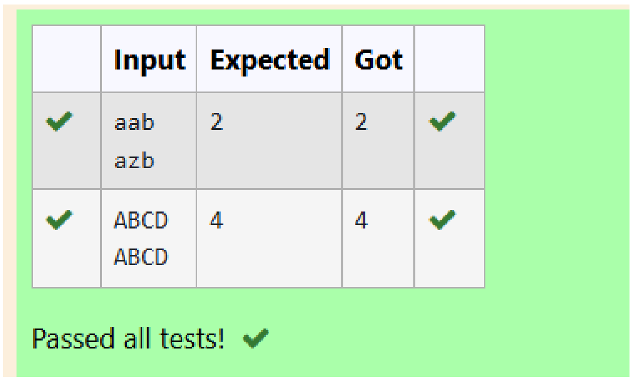
For example:

Input	Result
aab azb	2

Code:

```
#include <stdio.h>
#include <string.h>
#define MAX 100
int main() {
    char s1[MAX], s2[MAX];
    int a[MAX][MAX];
    scanf("%s", s1);
    scanf("%s", s2);
    int l1 = strlen(s1);
    int l2 = strlen(s2);
    for (int i = 0; i <= l1; i++) {
        for (int j = 0; j <= l2; j++) {
            if (i == 0 || j == 0) {
                a[i][j] = 0;
            } else if (s1[i - 1] == s2[j - 1]) {
                a[i][j] = a[i - 1][j - 1] + 1;
            } else {
                a[i][j] = a[i - 1][j] > a[i][j - 1] ? a[i - 1][j] : a[i][j - 1];
            }
        }
    }
    printf("%d\n",a[l1][l2]);
    return 0;
}
```

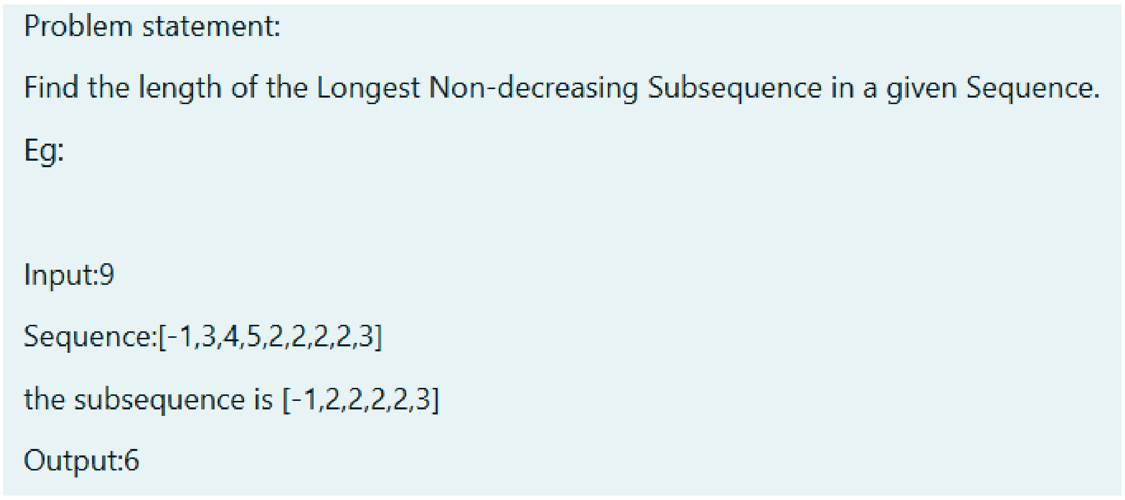
Output:



	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]

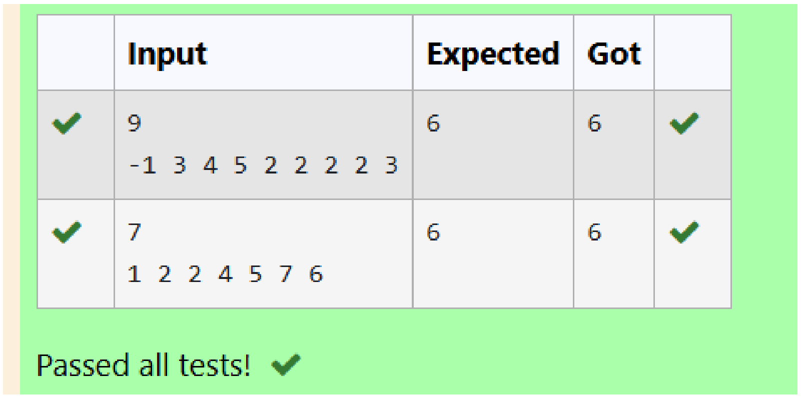
the subsequence is [-1,2,2,2,2,3]

Output:6

Code:

```
#include <stdio.h>
#define MAX 100
int main() {
    int n;
    int s[MAX];
    int a[MAX];
    scanf("%d",&n);
    for (int i=0;i<n;i++) {scanf("%d",&s[i]);}
    int m= 1;
    for (int i = 0; i < n; i++) {
        a[i] = 1;
        for (int j = 0; j < i; j++) {
            if (s[j] <= s[i]) {a[i] = a[i]>a[j]+1?a[i]:a[j]+1;}
        }
        if (a[i] > m) {m = a[i];}
    }
    printf("%d\n", m);
    return 0;
}
```

Output:



	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! ✓