

**NAME : KEERTHANA.P**

**REG NO: 241801122**

## **BASIC C PROGRAMMING-PRACTICE**

### **PROGRAM 1:**

**Question 1** | Correct Mark 1.00 out of 1.00 [Flag question](#)

Given two numbers, write a C program to swap the given numbers.

**For example:**

Input	Result
10 20	20 10

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

```
1 #include<stdio.h>
2 int main()
3 {
4     int a,b,c;
5     scanf("%d %d",&a,&b);
6     c=a;
7     a=b;
8     b=c;
9     printf("%d %d",a,b);
10 }
```

	Input	Expected	Got	
✓	10 20	20 10	20 10	✓

Passed all tests! ✓

## PROGRAM 2:

Question 2 | Correct Mark 1.00 out of 1.00 [Flag question](#)

Write a C program to find the eligibility of admission for a professional course based on the following criteria:

Marks in Maths  $\geq 65$

Marks in Physics  $\geq 55$

Marks in Chemistry  $\geq 50$

Or

Total in all three subjects  $\geq 180$

### Sample Test Cases

#### Test Case 1

##### Input

70 60 80

##### Output

The candidate is eligible

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

```
1 #include<stdio.h>
2 int main()
3 {
4     int a,b,c,t;
5     scanf("%d %d %d",&a,&b,&c);
6     t=a+b+c;
7     if((a>=65&&b>=55&&c>=50) || (t>=180))
8     {
9         printf("The candidate is eligible");
10    }
11    else
12    {
13        printf("The candidate is not eligible");
14    }
15 }
```

	Input	Expected	Got	
✓	70 60 80	The candidate is eligible	The candidate is eligible	✓
✓	50 80 80	The candidate is eligible	The candidate is eligible	✓

Passed all tests! ✓

## PROGRAM 3:

Question 3 | Correct Mark 1.00 out of 1.00 [Flag question](#)

Malini goes to BestSave hyper market to buy grocery items. BestSave hyper market provides 10% discount on the bill amount B when ever the bill amount B is more than Rs.2000.

The bill amount B is passed as the input to the program. The program must print the final amount A payable by Malini.

Input Format:

The first line denotes the value of B.

Output Format:

The first line contains the value of the final payable amount A.

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

```
1 #include<stdio.h>
2 int main()
3 {
4     int a,r;
5     scanf("%d",&a);
6     if(a<=2000)
7     {
8         r=a;
9     }
10    if(a>2000)
11    {
12        r=a-(a*0.1);
13    }
14    printf("%d",r);
15 }
```

	Input	Expected	Got	
✓	1900	1900	1900	✓
✓	3000	2700	2700	✓

Passed all tests! ✓

Correct

## PROGRAM 4:

Question 4 | Correct Mark 1.00 out of 1.00 [Flag question](#)

Baba is very kind to beggars and every day Baba donates half of the amount he has when ever a beggar requests him. The money M left in Baba's hand is passed as the input and the number of beggars B who received the alms are passed as the input. The program must print the money Baba had in the beginning of the day.

**Input Format:**

The first line denotes the value of M.  
The second line denotes the value of B.

**Output Format:**

The first line denotes the value of money with Baba in the beginning of the day.

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

```
1 #include<stdio.h>
2 int main()
3 {
4     int b,p,r;
5     scanf("%d %d",&b,&p);
6     r=b*(2*p);
7 }
8 }
```

	Input	Expected	Got	
✓	100 2	400	400	✓

Passed all tests! ✓

# PROGRAM 5:

Question 5 | Correct Mark 1.00 out of 1.00 [Flag question](#)

The CEO of company ABC Inc wanted to encourage the employees coming on time to the office. So he announced that for every consecutive day an employee comes on time in a week (starting from Monday to Saturday), he will be awarded Rs.200 more than the previous day as "Punctuality Incentive". The incentive I for the starting day (ie on Monday) is passed as the input to the program. The number of days N an employee came on time consecutively starting from Monday is also passed as the input. The program must calculate and print the "Punctuality Incentive" P of the employee.

#### Input Format:

The first line denotes the value of I.

The second line denotes the value of N.

#### Output Format:

The first line denotes the value of P.

```
-----  
1 #include<stdio.h>  
2 int main()  
3 {  
4     int a,b,r=0;  
5     scanf("%d %d",&a,&b);  
6     for(int i=0;i<b;i++)  
7     {  
8         r+=a+(200*i);  
9     }  
10    printf("%d",r);  
11 }
```

	Input	Expected	Got	
✓	500 3	2100	2100	✓
✓	100 3	900	900	✓

Passed all tests! ✓

# PROGRAM 6:

Question 6 | Correct Mark 1.00 out of 1.00 Flag question

Two numbers M and N are passed as the input. A number X is also passed as the input. The program must print the numbers divisible by X from N to M (inclusive of M and N).

**Input Format:**

The first line denotes the value of M  
The second line denotes the value of N  
The third line denotes the value of X

**Output Format:**

Numbers divisible by X from N to M, with each number separated by a space.

**Boundary Conditions:**

1 <= M <= 9999999  
M < N <= 9999999  
1 <= X <= 9999

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

```
1 #include<stdio.h>
2 int main()
3 {
4     int m,n,x;
5     scanf("%d %d %d",&m,&n,&x);
6     for(int i=n;i>=m;i--)
7     {
8         if(i%x==0)
9         {
10             printf("%d ",i);
11         }
12     }
13 }
```

	Input	Expected	Got	
✓	2	35 28 21 14 7	35 28 21 14 7	✓
	40			
	7			

Passed all tests! ✓

## PROGRAM 7:

Write a C program to find the quotient and remainder of given integers.

For example:

Input	Result
12	4
3	0

Answer: (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int a,b,q,r;
5     scanf("%d %d",&a,&b);
6     q=a/b;
7     r=a%b;
8     printf("%d\n%d",q,r);
9 }
```

	Input	Expected	Got	
✓	12	4	4	✓
	3	0	0	

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

## PROGRAM 8:

Write a C program to find the biggest among the given 3 integers?

For example:

Input	Result
10 20 30	30

Answer: (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int a,b,c,r;
5     scanf("%d %d %d",&a,&b,&c);
6     if(a>b&&a>c)
7     {
8         r=a;
9     }
10    else if(b>a&&b>c)
11    {
12        r=b;
13    }
14    else
15    {
16        r=c;
17    }
18    printf("%d",r);
19 }
```

	Input	Expected	Got	
✓	10 20 30	30	30	✓

Passed all tests! ✓

## PROGRAM 9:

Write a C program to find whether the given integer is odd or even?

For example:

Input	Result
12	Even
11	Odd

Answer: (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int a;
5     scanf("%d",&a);
6     if(a%2==0)
7     {
8         printf("Even");
9     }
10    else
11    {
12        printf("Odd");
13    }
14 }
```

	Input	Expected	Got	
✓	12	Even	Even	✓
✓	11	Odd	Odd	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

## PROGRAM 10:

Write a C program to find the factorial of given n.

For example:

Input	Result
5	120

Answer: (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int factorial(int n)
3 {
4     if(n==0 || n==1)
5     {
6         return 1;
7     }
8     return n*factorial(n-1);
9 }
10 int main()
11 {
12     int a,fact=0;
13     scanf("%d",&a);
14     fact+=factorial(a);
15     printf("%d",fact);
16 }
17 }
```

	Input	Expected	Got	
✓	5	120	120	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

## PROGRAM 11:

Write a C program to find the sum first N natural numbers.

For example:

Input	Result
3	6

Answer: (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int a,sum=0;
5     scanf("%d",&a);
6     for(int i=1;i<=a;i++)
7     {
8         sum+=i;
9     }
10    printf("%d",sum);
11 }
```

	Input	Expected	Got	
✓	3	6	6	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

## PROGRAM 12:

Question 12 | Correct Mark 1.00 out of 1.00 [Flag question](#)

Write a C program to find the Nth term in the fibonacci series.

For example:

Input	Result
0	0
1	1
4	3

Answer: (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main()
3 {
4     int n;
5     scanf("%d",&n);
6     int a=0,b=1,c;
7     for(int i=2;i<=n;i++)
8     {
9         c=a+b;
10        a=b;
11        b=c;
12    }
13    if(n==0)
14    {
15        c=a;
16    }
17    if(n==1)
18    {
19        c=b;
20    }
21    printf("%d",c);
22 }
```

	Input	Expected	Got	
✓	0	0	0	✓
✓	1	1	1	✓
✓	4	3	3	✓

Passed all tests! ✓

## PROGRAM 13:

Question 13 | Correct Mark 1.00 out of 1.00 [Flag question](#)

Write a C program to find the power of integers.

input:

a b

output:

$a^b$  value

For example:

Input	Result
2 5	32

Answer: (penalty regime: 0 %)

```
1 #include<stdio.h>
2 #include<math.h>
3 int main()
4 {
5     int a,b,r;
6     scanf("%d %d",&a,&b);
7     r=pow(a,b);
8     printf("%d",r);
9 }
```

	Input	Expected	Got	
✓	2 5	32	32	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

## PROGRAM 14:

Question 14 | Correct Mark 1.00 out of 1.00 [Flag question](#)

Write a C program to find Whether the given integer is prime or not.

For example:

Input	Result
7	Prime
9	No Prime

Answer: (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int prime(int n,int d)
3 {
4     if(n<=1)
5     {
6         return 0;
7     }
8     if(d==1)
9     {
10        return 1;
11    }
12    if(n%d==0)
13    {
14        return 0;
15    }
16    return prime(n,d-1);
17 }
18 int main()
19 {
20     int n;
21     scanf("%d",&n);
22     if(prime(n,n/2))
23     {
24         printf("Prime");
25     }
26     else
27     {
28         printf("No Prime");
29     }
30 }
```

	Input	Expected	Got	
✓	7	Prime	Prime	✓
✓	9	No Prime	No Prime	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

## PROGRAM 15:

Question 15 | Correct Mark 1.00 out of 1.00 [Flag question](#)

Write a C program to find the reverse of the given integer?

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

```
1 #include<stdio.h>
2 int main()
3 {
4     int a;
5     scanf("%d",&a);
6     int digit,rev=0;
7     while(a!=0)
8     {
9         digit=a%10;
10        rev=(rev*10)+digit;
11        a/=10;
12    }
13    printf("%d",rev);
14 }
```

	Input	Expected	Got	
✓	123	321	321	✓

Passed all tests! ✓

**Correct**

Marks for this submission: 1.00/1.00.

**NAME: KEERTHANA.P**

**REG NO: 241801122**

## **GREEDY ALGORITHMS**

### **PROGRAM 1:**

Question 1 | Correct Mark 1.00 out of 1.00 [Flag question](#)

Write a program to take value V and we want to make change for V Rs, and we have infinite supply of each of the denominations in Indian currency, i.e., we have infinite supply of { 1, 2, 5, 10, 20, 50, 100, 500, 1000} valued coins/notes, what is the minimum number of coins and/or notes needed to make the change.

Input Format:

Take an integer from stdin.

Output Format:

print the integer which is change of the number.

Example Input :

64

Output:

4

Explanation:

We need a 50 Rs note and a 10 Rs note and two 2 rupee coins.

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

```
1 #include <stdio.h>
2 int main()
3 {
4     int v;
5     scanf("%d", &v);
6     int denominations[] = {1000, 500, 100, 50, 20, 10, 5, 2, 1};
7     int n = sizeof(denominations) / sizeof(denominations[0]);
8     int count = 0;
9     for (int i = 0; i < n; i++)
10    {
11        if (v >= denominations[i])
12        {
13            count += v / denominations[i];
14            v = v % denominations[i];
15        }
16    }
17
18    printf("%d\n", count);
19
20 }
```

	Input	Expected	Got	
✓	49	5	5	✓

Passed all tests! ✓

## PROGRAM 2:

Question 1 | Correct Mark 1.00 out of 1.00 [Flag question](#)

Assume you are an awesome parent and want to give your children some cookies. But, you should give each child at most one cookie.

Each child  $i$  has a greed factor  $g[i]$ , which is the minimum size of a cookie that the child will be content with; and each cookie  $j$  has a size  $s[j]$ . If  $s[j] \geq g[i]$ , we can assign the cookie  $j$  to the child  $i$ , and the child  $i$  will be content. Your goal is to maximize the number of your content children and output the maximum number.

**Example 1:**

**Input:**

3

1 2 3

2

1 1

**Output:**

1

Explanation: You have 3 children and 2 cookies. The greed factors of 3 children are 1, 2, 3.

And even though you have 2 cookies, since their size is both 1, you could only make the child whose greed factor is 1 content.

You need to output 1.

**Constraints:**

$1 \leq g.length \leq 3 * 10^4$

$0 \leq s.length \leq 3 * 10^4$

$1 \leq g[i], s[j] \leq 2^{31} - 1$

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

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int cmp(const void *a, const void *b)
5 {
6     int x = *(int*)a;
7     int y = *(int*)b;
8     return (x-y);
9 }
10
11 int main() {
12     int n, m;
13     scanf("%d", &n);
14     int g[n];
15     for (int i = 0; i < n; i++)
16     {
17         scanf("%d", &g[i]);
18     }
19     scanf("%d", &m);
20     int s[m];
21     for (int i = 0; i < m; i++)
22     {
23         scanf("%d", &s[i]);
24     }
25     qsort(g, n, sizeof(int), cmp);
26     qsort(s, m, sizeof(int), cmp);
27     int i = 0, j = 0;
28     int content_children = 0;
29     while (i < n && j < m)
30     {
31         if (s[j] >= g[i])
32         {
33             content_children++;
34             i++;
35             j++;
36         }
37         else
38         {
39             j++;
40         }
41     }
42     printf("%d\n", content_children);
43     return 0;
44 }
45
```

	Input	Expected	Got	
✓	2	2	2	✓
	1 2			
	3			
	1 2 3			

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

# PROGRAM 3:

Question 1 | Correct Mark 1.00 out of 1.00 ⚡ [Flag question](#)

A person needs to eat burgers. Each burger contains a count of calorie. After eating the burger, the person needs to run a distance to burn out his calories. If he has eaten  $i$  burgers with  $c$  calories each, then he has to run at least  $3^i * c$  Kilometers to burn out the calories. For example, if he ate 3 burgers with the count of calorie in the order: [1, 3, 2], the kilometers he needs to run are  $(3^0 * 1) + (3^1 * 3) + (3^2 * 2) = 1 + 9 + 18 = 28$ . But this is not the minimum, so need to try out other orders of consumption and choose the minimum value. Determine the minimum distance he needs to run. Note: He can eat burger in any order and use an efficient sorting algorithm. Apply greedy approach to solve the problem.

## Input Format

First Line contains the number of burgers

Second line contains calories of each burger which is  $n$  space-separate integers

## Output Format

Print: Minimum number of kilometers needed to run to burn out the calories

## Sample Input

3

5 10 7

## Sample Output

76

## For example:

Test	Input	Result
Test Case 1	3 1 3 2	18

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

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<math.h>
4 int cmp_desc(const void *a, const void *b)
5 {
6     return (*(int*)b - *(int*)a);
7 }
8
9 int main() {
10     int n;
11     scanf("%d", &n);
12     int a[n];
13     for (int i = 0; i < n; i++) {
14         scanf("%d", &a[i]);
15     }
16     qsort(a, n, sizeof(int), cmp_desc);
17     int c=0;
18     for(int i=0;i<n;i++)
19     {
20         c+=(pow(n,i)*a[i]);
21     }
22     printf("%d",c);
23 }
```

	Test	Input	Expected	Got	
✓	Test Case 1	3 1 3 2	18	18	✓
✓	Test Case 2	4 7 4 9 6	389	389	✓
✓	Test Case 3	3 5 10 7	76	76	✓

Passed all tests! ✓

**Correct**

Marks for this submission: 1.00/1.00.

## PROGRAM 4:

Question 1 | Correct Mark 1.00 out of 1.00 [Flag question](#)

Given an array of N integer, we have to maximize the sum of arr[i] \* i, where i is the index of the element (i = 0, 1, 2, ..., N). Write an algorithm based on Greedy technique with a Complexity O(nlogn).

Input Format:

First line specifies the number of elements-n

The next n lines contain the array elements.

Output Format:

Maximum Array Sum to be printed.

Sample Input:

5

2 5 3 4 0

Sample output:

40

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

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 int cmp(const void *a, const void *b)
4 {
5     return (*(int*)a - *(int*)b);
6 }
7
8 int main()
9 {
10    int n;
11    scanf("%d",&n);
12    int a[n],sum=0;
13    for(int i=0;i<n;i++)
14    {
15        scanf("%d",&a[i]);
16    }
17    qsort(a,n,sizeof(int),cmp);
18    for(int i=0;i<n;i++)
19    {
20        sum+=a[i]*i;
21    }
22    printf("%d",sum);
23 }
24 }
```

	Input	Expected	Got	
✓	5 2 5 3 4 0	40	40	✓
✓	10 2 2 2 4 4 3 3 5 5 5	191	191	✓
✓	2 45 3	45	45	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

## PROGRAM 5:

Question 1 | Correct Mark 1.00 out of 1.00 [Flag question](#)

Given two arrays array\_One[] and array\_Two[] of same size N. We need to first rearrange the arrays such that the sum of the product of pairs( 1 element from each) is minimum. That is  $\text{SUM } (A[i] * B[i])$  for all i is minimum.

For example:

Input	Result
3	28
1	
2	
3	
4	
5	
6	

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

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 int asc(const void *a, const void *b)
4 {
5     return (*(int*)a - *(int*)b);
6 }
7 int desc(const void *a, const void *b) {
8     return (*(int*)b - *(int*)a);
9 }
10 int main()
11 {
12     int n;
13     scanf("%d",&n);
14     int a[n],b[n],sum=0;
15     for(int i=0;i<n;i++)
16     {
17         scanf("%d",&a[i]);
18     }
19     for(int i=0;i<n;i++)
20     {
21         scanf("%d",&b[i]);
22     }
23     qsort(a,n,sizeof(int),asc);
24     qsort(b,n,sizeof(int),desc);
25     for(int i=0;i<n;i++)
26     {
27         sum+=a[i]*b[i];
28     }
29     printf("%d",sum);
30 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	3 1 2 3 4 5 6	28	28	✓
✓	4 7 5 1 2 1 3 4 1	22	22	✓
✓	5 20 10 30 10 40 8 9 4 3 10	590	590	✓

**NAME: KEERTHANA.P**

**REG NO: 241801122**

## **COMPETITIVE PROGRAMMING**

### **PROGRAM 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 1 2 3 4	

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

```
1 #include <stdio.h>
2 int main()
3 {
4     int n;
5     scanf("%d", &n);
6     int arr[n];
7     for (int i = 0; i < n; i++)
8     {
9         scanf("%d", &arr[i]);
10    }
11    int duplicate = -1;
12    for (int i = 0; i < n; i++)
13    {
14        for (int j = i + 1; j < n; j++)
15        {
16            if (arr[i] == arr[j]) {
17                duplicate = arr[i];
18                break;
19            }
20        }
21        if (duplicate != -1)
22            break;
23    }
24    if (duplicate != -1)
25        printf("%d\n", duplicate);
26    else
27        printf("No duplicate found\n");
28    return 0;
29 }
30
31 }
```

	Input	Expected	Got	
✓	11 10 9 7 6 5 1 2 3 8 4 7	7	7	✓
✓	5 1 2 3 4 4	4	4	✓
✓	5 1 1 2 3 4	1	1	✓

Passed all tests! ✓

**Correct**

Marks for this submission: 1.00/1.00.

## **PROGRAM 2: Finding Duplicates-O(n) 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 1 2 3 4	

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

```
1 #include <stdio.h>
2 int main()
3 {
4     int n;
5     scanf("%d", &n);
6     int arr[n];
7     for (int i = 0; i < n; i++)
8         scanf("%d", &arr[i]);
9     int slow = arr[0];
10    int fast = arr[0];
11    do
12    {
13        slow = arr[slow];
14        fast = arr[arr[fast]];
15    } while (slow != fast);
16    slow = arr[0];
17    while (slow != fast)
18    {
19        slow = arr[slow];
20        fast = arr[fast];
21    }
22    printf("%d\n", slow);
23    return 0;
24 }
```

	Input	Expected	Got	
✓	11 10 9 7 6 5 1 2 3 8 4 7	7	7	✓
✓	5 1 2 3 4 4	4	4	✓
✓	5 1 1 2 3 4	1	1	✓

Passed all tests! ✓

## **PROGRAM 3: Print Intersection of 2 sorted arrays-O(m\*n)Time Complexity,O(1) Space Complexity**

---

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

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

```
1 #include <stdio.h>
2 int main()
3 {
4     int t;
5     scanf("%d", &t);
6     while (t--)
7     {
8         int n1, n2;
9         scanf("%d", &n1);
10        int arr1[n1];
11        for (int i = 0; i < n1; i++)
12            scanf("%d", &arr1[i]);
13        scanf("%d", &n2);
14        int arr2[n2];
15        for (int i = 0; i < n2; i++)
16            scanf("%d", &arr2[i]);
17        for (int i = 0; i < n1; i++)
18        {
19            for (int j = 0; j < n2; j++)
20            {
21                if (arr1[i] == arr2[j])
22                {
23                    printf("%d ", arr1[i]);
24                    break;
25                }
26            }
27        }
28        printf("\n");
29    }
30    return 0;
31 }
32 }
```

	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! ✓

Correct

Marks for this submission: 1.00/1.00.

## **PROGRAM 4: Print Intersection of 2 sorted arrays-O(m+n)Time Complexity,O(1) Space Complexity**

**Question 1** | Correct Mark 1.00 out of 1.00  [Flag question](#)

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

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

```
1 #include <stdio.h>
2 int main()
3 {
4     int t;
5     scanf("%d", &t);
6     while (t--)
7     {
8         int n1, n2;
9         scanf("%d", &n1);
10        int arr1[n1];
11        for (int i = 0; i < n1; i++)
12            scanf("%d", &arr1[i]);
13        scanf("%d", &n2);
14        int arr2[n2];
15        for (int i = 0; i < n2; i++)
16            scanf("%d", &arr2[i]);
17        int i = 0, j = 0;
18        while (i < n1 && j < n2)
19        {
20            if (arr1[i] < arr2[j])
21                i++;
22            else if (arr2[j] < arr1[i])
23                j++;
24            else {
25                printf("%d ", arr1[i]);
26                i++;
27                j++;
28            }
29        }
30        printf("\n");
31    }
32    return 0;
33 }
```

	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! ✓

Correct

Marks for this submission: 1.00/1.00.

# PROGRAM 5: Pair with Difference-O(n^2)Time Complexity,O(1) Space Complexity

Question 1 | Correct Mark 1.00 out of 1.00 [Flag question](#)

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

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.

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

```
1 #include<stdio.h>
2 int main()
3 {
4     int n;
5     scanf("%d",&n);
6     int arr[n];
7     for(int i=0;i<n;i++)
8     {
9         scanf("%d",&arr[i]);
10    }
11    int k;
12    scanf("%d",&k);
13    int found=0;
14    for(int i=0;i<n;i++)
15    {
16        for(int j=i+1;j<n;j++)
17        {
18            if(arr[j]-arr[i]==k)
19            {
20                found=1;
21                break;
22            }
23        }
24        if(found) break;
25    }
26    printf("%d\n", found);
27    return 0;
28 }
29 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	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! ✓

**Correct**

Marks for this submission: 1.00/1.00.

# PROGRAM 6: Pair with Difference -O(n) Time Complexity,O(1) Space Complexity

Question 1 | Correct Mark 1.00 out of 1.00 [Flag question](#)

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

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
1 3 5	
4	

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

```
1 #include <stdio.h>
2 int main()
3 {
4     int n;
5     scanf("%d", &n);
6     int arr[n];
7     for (int i = 0; i < n; i++)
8         scanf("%d", &arr[i]);
9     int k;
10    scanf("%d", &k);
11    int i = 0, j = 1;
12    int found = 0;
13    while (i < n && j < n)
14    {
15        int diff = arr[j] - arr[i];
16        if (i != j && diff == k)
17        {
18            found = 1;
19            break;
20        }
21        else if (diff < k)
22            j++;
23        else
24            i++;
25    }
26    printf("%d\n", found);
27    return 0;
28 }
```

	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! ✓

Correct

Marks for this submission: 1.00/1.00.

**NAME: KEERTHANA.P**

**REG NO: 241801122**

## **FINDING TIME COMPLEXITY OF ALGORITHMS**

### **PROGRAM 1:**

**Question 1** | Correct Mark 1.00 out of 1.00 [Flag question](#)

Convert the following algorithm into a program and find its time complexity using the counter method.

```
void function (int n)
{
    int i= 1;

    int s =1;

    while(s <= n)
    {
        i++;
        s += i;
    }
}
```

**Note:** No need of counter increment for declarations and scanf() and count variable printf() statements.

**Input:**

A positive Integer n

**Output:**

Print the value of the counter variable

**For example:**

Input	Result
9	12

Ace editor not ready. Perhaps reload page?

Falling back to raw text area.

---

```
#include<stdio.h>
void function(int n)
{
    int count=0;
    int i=1;
    count++;
    int s=1;
    count++;
    while(s<=n)
    {
        count++;
        i++;
        count++;
        s+=i;
        count++;
    }
    count++;
    printf("%d", count);
```

---

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	9	12	12	✓
✓	4	9	9	✓

Passed all tests! ✓

## PROGRAM 2:

Convert the following algorithm into a program and find its time complexity using the counter method.

```
void func(int n)
{
    if(n==1)
    {
        printf("*");
    }
    else
    {
        for(int i=1; i<=n; i++)
        {
            for(int j=1; j<=n; j++)
            {
                printf("*");
                printf("*");
                break;
            }
        }
    }
}
```

**Note:** No need of counter increment for declarations and scanf() and count variable printf() statements.

**Input:**

A positive Integer n

**Output:**

Print the value of the counter variable

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

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

	Input	Expected	Got	
✓	2	12	12	✓
✓	1000	5002	5002	✓
✓	143	717	717	✓

Passed all tests! ✓

Correct

## PROGRAM 3:

Question 1 | Correct Mark 1.00 out of 1.00 [Flag question](#)

Convert the following algorithm into a program and find its time complexity using counter method.

```
Factor(num) {  
    {  
        for (i = 1; i <= num; ++i)  
        {  
            if (num % i == 0)  
            {  
                printf("%d ", i);  
            }  
        }  
    }  
}
```

**Note:** No need of counter increment for declarations and scanf() and counter variable printf() statement.

**Input:**

A positive Integer n

**Output:**

Print the value of the counter variable

**Answer:**

```
1 #include<stdio.h>
2 void Factor(int num)
3 {
4     int count=0;
5     for (int i = 1; i <= num; ++i)
6     {
7         count++;
8         if (num % i== 0)
9         {
10             count++;
11             //printf("%d ", i);
12         }
13         count++;
14     }
15     count++;
16     printf("%d",count);
17 }
18 int main()
19 {
20     int n;
21     scanf("%d",&n);
22     Factor(n);
23 }
```

	Input	Expected	Got	
✓	12	31	31	✓
✓	25	54	54	✓
✓	4	12	12	✓

## PROGRAM 4:

Question 1 | Correct Mark 1.00 out of 1.00 [Flag question](#)

Convert the following algorithm into a program and find its time complexity using counter method.

```
void reverse(int n)
{
    int rev = 0, remainder;
    while (n != 0)
    {
        remainder = n % 10;
        rev = rev * 10 + remainder;
        n/= 10;

    }
    print(rev);
}
```

**Note:** No need of counter increment for declarations and scanf() and count variable printf() statements.

**Input:**

A positive Integer n

**Output:**

Print the value of the counter variable

```
1 #include<stdio.h>
2 void reverse(int n)
3 {
4     int count=0;
5     int rev = 0, remainder;
6     count++;
7     count++;
8     while (n != 0)
9     {
10         count++;
11         remainder = n % 10;
12         count++;
13         rev = rev * 10 + remainder;
14         count++;
15         n/= 10;
16         count++;
17     }
18     count++;
19     //print(rev);
20     printf("%d",count);
21 }
22 int main()
23 {
24     int n;
25     scanf("%d",&n);
26     reverse(n);
27 }
```

	Input	Expected	Got	
✓	12	11	11	✓
✓	1234	19	19	✓

Passed all tests! ✓

**NAME: KEERTHANA.P**

**REG NO: 241801122**

## **DYNAMIC PROGRAMMING**

### **PROGRAM 1:**

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 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 long long countWays(int n) {
4     if (n < 0) return 0;
5     if (n == 0) return 1;
6
7     long long dp[n + 1];
8     dp[0] = 1;
9
10    for (int i = 1; i <= n; i++) {
11        dp[i] = 0;
12        if (i - 1 >= 0)
13            dp[i] += dp[i - 1];
14        if (i - 3 >= 0)
15            dp[i] += dp[i - 3];
16    }
17
18    return dp[n];
19}
20
21 int main() {
22     int n;
23     scanf("%d", &n);
24     printf("%lld", countWays(n));
25     return 0;
26 }
27
```

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

Passed all tests! ✓

## PROGRAM 2:

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 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  $n \times n$  chessboard values

### Output Format

Print Maximum monetary value of the path

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

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

	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.

## PROGRAM 3:

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	2
azb	

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

```
1 #include <stdio.h>
2 #include <string.h>
3 int max(int a, int b)
4 {
5     return (a > b) ? a : b;
6 }
7 int main()
8 {
9     char s1[1000], s2[1000];
10    scanf("%s %s", s1, s2);
11    int m = strlen(s1);
12    int n = strlen(s2);
13    int dp[m+1][n+1];
14    for (int i = 0; i <= m; i++)
15    {
16        for (int j = 0; j <= n; j++)
17        {
18            if (i == 0 || j == 0)
19                dp[i][j] = 0;
20            else if (s1[i-1] == s2[j-1])
21                dp[i][j] = dp[i-1][j-1] + 1;
22            else
23                dp[i][j] = max(dp[i-1][j], dp[i][j-1]);
24        }
25    }
26    printf("%d\n", dp[m][n]);
27    return 0;
28 }
```

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

Passed all tests! ✓

## PROGRAM 4:

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

	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! ✓

Correct

Marks for this submission: 1.00/1.00.

**NAME: KEERTHANA.P**

**REG NO: 241801122**

## **DIVIDE AND CONQUER**

### **PROGRAM 1:**

Question 1 | Correct Mark 1.00 out of 1.00 [Flag question](#)

#### **Problem Statement**

Given an array of 1s and 0s this has all 1s first followed by all 0s. Aim is to find the number of 0s. Write a program using Divide and Conquer to Count the number of zeroes in the given array.

#### **Input Format**

First Line Contains Integer m – Size of array

Next m lines Contains m numbers – Elements of an array

#### **Output Format**

First Line Contains Integer – Number of zeroes present in the given array.

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

```
1 #include <stdio.h>
2
3 int find_first_zero(int arr[], int low, int high) {
4     int result = -1;
5     while (low <= high) {
6         int mid = (low + high) / 2;
7         if (arr[mid] == 0) {
8             result = mid;
9             high = mid - 1;
10        } else {
11            low = mid + 1;
12        }
13    }
14    return result;
15 }
16
17 int main() {
18     int m;
19     scanf("%d", &m);
20
21     int arr[m];
22     for (int i = 0; i < m; i++) {
23         scanf("%d", &arr[i]);
24     }
25
26     int first_zero_index = find_first_zero(arr, 0, m - 1);
27
28     if (first_zero_index == -1) {
29         printf("0\n");
30     } else {
31         printf("%d", m - first_zero_index);
32     }
33
34     return 0;
35 }
36 }
```

	Input	Expected	Got	
✓	5 1 1 1 0 0	2	2	✓
✓	10 1 1 1 1 1 1 1 1	0	0	✓
✓	8 0 0 0 0 0 0 0 0	8	8	✓
✓	17 1 1 1 1 1 1 1 1 1 1 1 1 0 0	2	2	✓

## PROGRAM 2:

Question 1 | Correct Mark 1.00 out of 1.00 [Flag question](#)

Given an array `nums` of size `n`, return *the majority element*.

The majority element is the element that appears more than  $\lfloor n / 2 \rfloor$  times. You may assume that the majority element always exists in the array.

**Example 1:**

Input: `nums = [3,2,3]`

Output: 3

**Example 2:**

Input: `nums = [2,2,1,1,1,2,2]`

Output: 2

**Constraints:**

- `n == nums.length`
- `1 <= n <= 5 * 104`
- `-231 <= nums[i] <= 231 - 1`

**For example:**

Input	Result
3	3
3 2 3	
7	2
2 2 1 1 1 2 2	

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

```
1 #include <stdio.h>
2
3 int majorityElement(int nums[], int numsSize) {
4     int count = 0, candidate = 0;
5
6     for (int i = 0; i < numsSize; i++) {
7         if (count == 0) {
8             candidate = nums[i];
9             count = 1;
10        } else if (nums[i] == candidate) {
11            count++;
12        } else {
13            count--;
14        }
15    }
16    return candidate;
17}
18
19 int main() {
20     int n;
21     scanf("%d", &n);
22
23     int nums[n];
24     for (int i = 0; i < n; i++) {
25         scanf("%d", &nums[i]);
26     }
27
28     int result = majorityElement(nums, n);
29     printf("%d\n", result);
30
31     return 0;
32}
33
```

	Input	Expected	Got	
✓	3 3 2 3	3	3	✓

Passed all tests! ✓

# PROGRAM 3:

Question 1 | Correct Mark 1.00 out of 1.00 [Flag question](#)

## Problem Statement:

Given a sorted array and a value x, the floor of x is the largest element in array smaller than or equal to x. Write divide and conquer algorithm to find floor of x.

## Input Format

First Line Contains Integer n – Size of array

Next n lines Contains n numbers – Elements of an array

Last Line Contains Integer x – Value for x

## Output Format

First Line Contains Integer – Floor value for x

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

```
1 #include <stdio.h>
2
3 int findFloor(int arr[], int n, int x) {
4     int low = 0, high = n - 1;
5     int floor = -1;
6
7     while (low <= high) {
8         int mid = low + (high - low) / 2;
9
10        if (arr[mid] == x) {
11            return arr[mid];
12        }
13        else if (arr[mid] < x) {
14            floor = arr[mid];
15            low = mid + 1;
16        }
17        else {
18            high = mid - 1;
19        }
20    }
21
22    return floor;
23 }
24
25 int main() {
26     int n, x;
27     scanf("%d", &n);
28     int arr[n];
29
30     for (int i = 0; i < n; i++) {
31         scanf("%d", &arr[i]);
32     }
33     scanf("%d", &x);
34
35     int result = findFloor(arr, n, x);
36     printf("%d\n", result);
37
38     return 0;
39 }
40 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	6 1 2 8 10 12 19 5	2	2	✓
✓	5 10 22 85 108 129 100	85	85	✓
✓	7 3 5 7 9 11 13 15 10	9	9	✓

Passed all tests! ✓

# PROGRAM 4:

Question 1 | Correct Mark 1.00 out of 1.00 [Flag question](#)

## Problem Statement:

Given a sorted array of integers say arr[] and a number x. Write a recursive program using divide and conquer strategy to check if there exist two elements in the array whose sum = x. If there exist such two elements then return the numbers, otherwise print as "No".

Note: Write a Divide and Conquer Solution

## Input Format

First Line Contains Integer n – Size of array

Next n lines Contains n numbers – Elements of an array

Last Line Contains Integer x – Sum Value

## Output Format

First Line Contains Integer – Element1

Second Line Contains Integer – Element2 (Element 1 and Elements 2 together sums to value "x")

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

```
1 #include <stdio.h>
2
3 int elem1 = 0, elem2 = 0;
4
5 int findPair(int arr[], int left, int right, int x) {
6     if (left >= right) {
7         return 0;
8     }
9
10    int sum = arr[left] + arr[right];
11    if (sum == x) {
12        elem1 = arr[left];
13        elem2 = arr[right];
14        return 1;
15    }
16    else if (sum < x) {
17        return findPair(arr, left + 1, right, x);
18    }
19    else {
20        return findPair(arr, left, right - 1, x);
21    }
22}
23
24 int main() {
25     int n, x;
26     scanf("%d", &n);
27     int arr[n];
28
29     for (int i = 0; i < n; i++) {
30         scanf("%d", &arr[i]);
31     }
32     scanf("%d", &x);
33
34     if (findPair(arr, 0, n - 1, x)) {
35         printf("%d\n%d\n", elem1, elem2);
36     } else {
37         printf("No\n");
38     }
39
40     return 0;
41 }
42 }
```

	Input	Expected	Got	
✓	4	4	4	✓
	2	10	10	
	4			
	8			
	10			
	14			

  

✓	5	No	No	✓
	2			
	4			
	6			
	8			
	10			
	100			

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

## PROGRAM 5:

Question 1 | Correct Mark 1.00 out of 1.00  Flag question

Write a Program to Implement the Quick Sort Algorithm

Input Format:

The first line contains the no of elements in the list-n

The next n lines contain the elements.

Output:

Sorted list of elements

For example:

Input	Result
5	12 34 67 78 98
67 34 12 98 78	

**Answer:**

```
1 #include <stdio.h>
2
3 void swap(int *a, int *b) {
4     int temp = *a;
5     *a = *b;
6     *b = temp;
7 }
8
9 int partition(int arr[], int low, int high) {
10    int pivot = arr[high];
11    int i = low - 1;
12
13    for (int j = low; j <= high - 1; j++) {
14        if (arr[j] <= pivot) {
15            i++;
16            swap(&arr[i], &arr[j]);
17        }
18    }
19    swap(&arr[i + 1], &arr[high]);
20    return i + 1;
21 }
22
23 void quickSort(int arr[], int low, int high) {
24    if (low < high) {
25        int pi = partition(arr, low, high);
26
27        quickSort(arr, low, pi - 1);
28        quickSort(arr, pi + 1, high);
29    }
30 }
31
32 int main() {
33    int n;
34    scanf("%d", &n);
35
36    int arr[n];
37    for (int i = 0; i < n; i++) {
38        scanf("%d", &arr[i]);
39    }
40
41    quickSort(arr, 0, n - 1);
42
43    for (int i = 0; i < n; i++) {
44        printf("%d ", arr[i]);
45    }
46    printf("\n");
47
48    return 0;
49 }
50
```

	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 56 11 10 90 114	1 10 11 32 56 56 78 90 90 114	1 10 11 32 56 56 78 90 90 114	✓
✓	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 90	✓

Passed all tests! ✓

**Correct**

Marks for this submission: 1.00/1.00.