RAJALAKSHMI ENGINEERIN GCOLLEGE

An Autonomous Institution, Affiliated to Anna University Rajalakshmi Nagar, Thandalam – 602 105



DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

CS23331- DESIGN AND ANALYSIS OF ALGORITHM (Regulation 2023)

LAB RECORD

Name:
Register Number:
Year/Branch/Section:
Semester:
Academic Year:

RAJALAKSHMI ENGINEERING COLLEGE RAJALAKSHMI NAGAR, THANDALAM-602 105

BONAFIDE CERTIFICATE

NAME	
ACADEMIC YEAR SEM	1ESTER
BRANCH	••••••
UNIVERSITY REGISTER NO:	
in the	ord of work done by the above student
during the year 2020	Laboratory
	Signature of Faculty-in-charge
Submitted for the Practical Exam	mination held on
Internal Examiner	External Examiner

CONTENT

S.No	Date	Name of the experiments	Page No	Faculty sign
1.		Basics of C programming		
2.		Finding Time Complexity of algorithms		
3.		Design and implement algorithms using Divide and Conquer Technique		
4.		Design and implement algorithms using Greedy Technique		
5.		Design and implement algorithms using Dynamic Programming		
6.		Competitive Programming-Certain Techniques		

EXP.NO:1(a)	DAGIC C DDOCDAMMING DDAGGG
DATE:	BASIC C PROGRAMMING-PRACTICE
<u>M</u> :	
.GORITHM:	

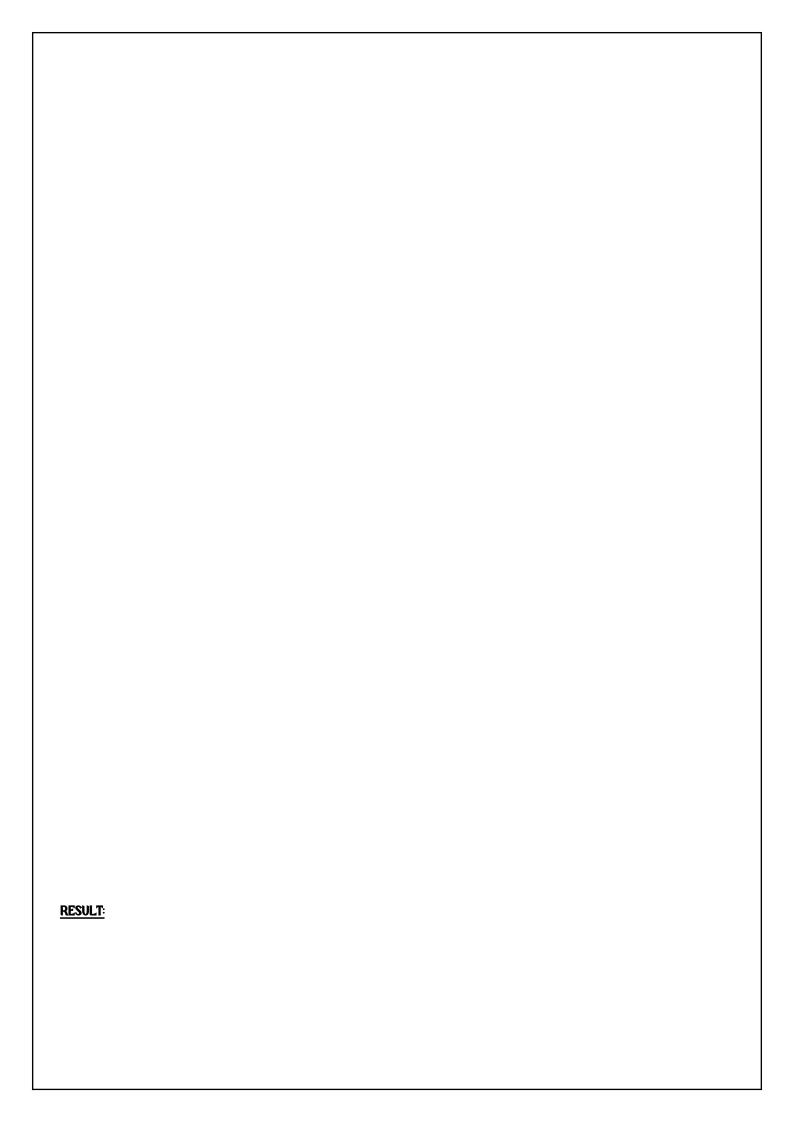
Given two numbers, write a C program to swap the given numbers.

For example:

Input	Result
10 20	20 10

```
#include<stdio.h>
int main()
(
int a, b, t:
    scanf("%d %d", &a, &b):
    t=a:
    a=b:
    b=t:
    printf("%d %d", a, b):
    return O:
)
```

	Input	Expected	Got	
~	10 20	20 10	20 10	~



DASIC C P	
DATE:	ROGRAMMING-PRACTICE

AIM:

ALGORITHM:

```
Write a \mbox{\it C} program to find the eligibility of admission for a professional course based on the following criteria:
```

```
Marks in Maths >= 65
```

Marks in Physics >= 55

Marks in Chemistry >= 50

Or

Total in all three subjects >= 180

Sample Test Cases

Test Case 1

Input

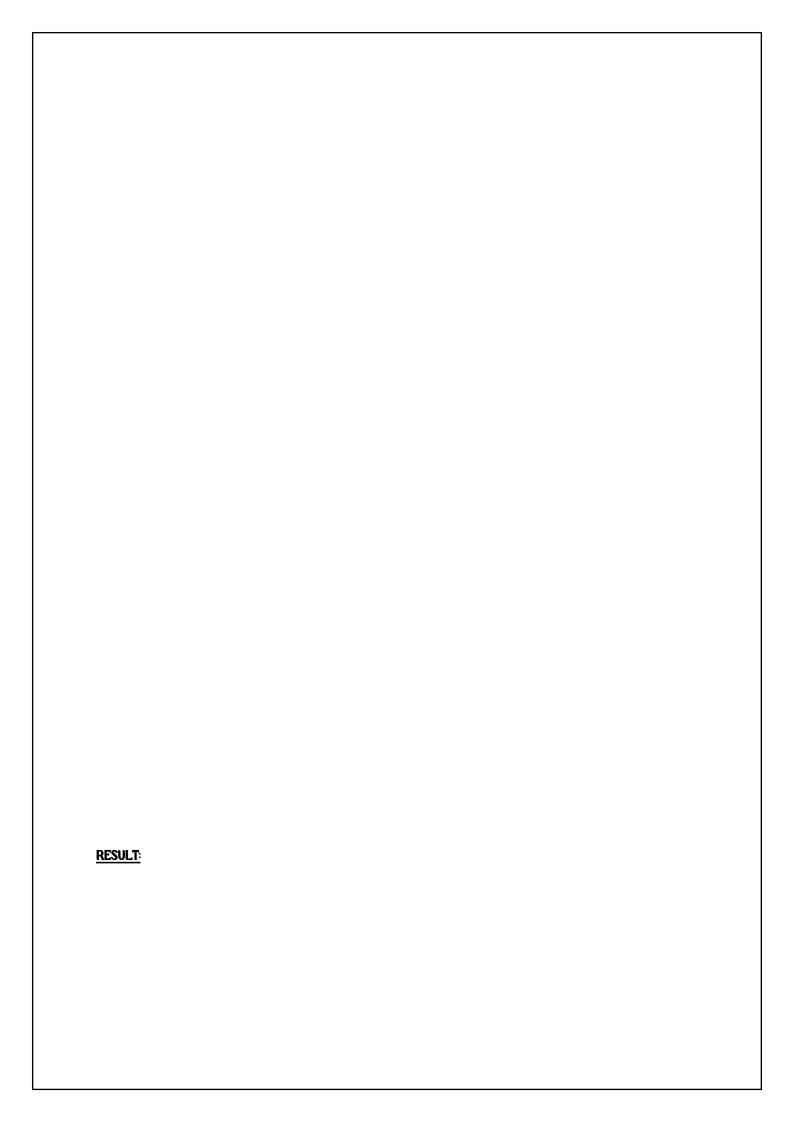
70 60 80

Output

The candidate is eligible

```
#include<stdio.h>
int main()
{
    int m.p.c:
    scanf("%d %d %d",&m,&p,&c):
    int t=m+p+c:
    if(m>=65 && p>=55 && c>=50)(
        printf("The candidate is eligible"):
    }
    else if(t>=180)(
        printf("The candidate is eligible"):
    }
    else(
        printf("The candidate is not eligible"):
    }
}
```

	Inpu	ıt		Expected	Got	
~	70	60	80	The candidate is eligible	The candidate is eligible	~
~	50 8	0 80		The candidate is eligible	The candidate is eligible	~



EXP.NO:1(c)	PAGIC C DESCRAMMING PRACTICE
DATE:	BASIC C PROGRAMMING-PRACTICE
<u>M</u> :	
_GORITHM:	

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.

Example Input/Output 1:

Input:

1900

Output:

1900

Example Input/Output 2:

Input:

3000

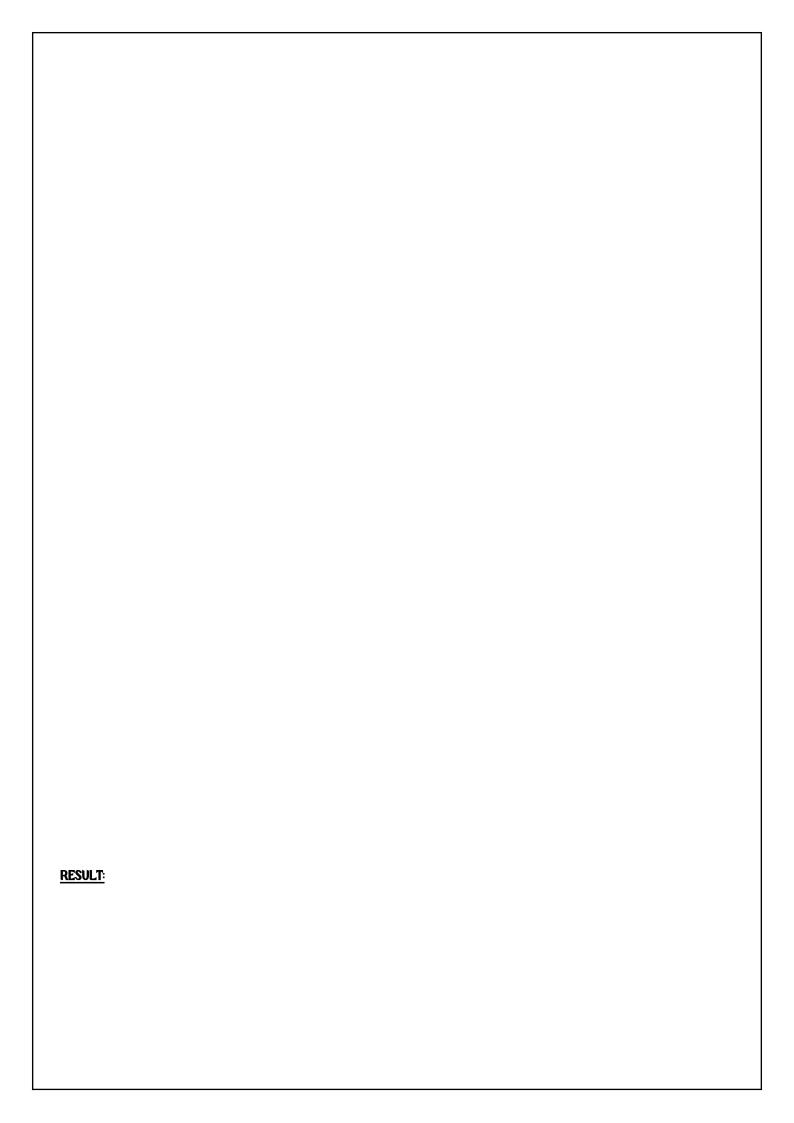
Output:

2700

```
#include<stdio.h>
int main(){

int c, t:
    scanf("%d",&c):
    if(c>2000)(
        t=c-(c'0.1):
    }
    else{
        t=c:
    }
    printf("%d",t):
}
```

	Input	Expected	Got	
~	1900	1900	1900	~
~	3000	2700	2700	~



EXP.NO:1(d)	
DATE:	BASIC C PROGRAMMING-PRACTICE
<u>M</u> :	
GORITHM:	

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.

Example Input/Output:

Input:

100

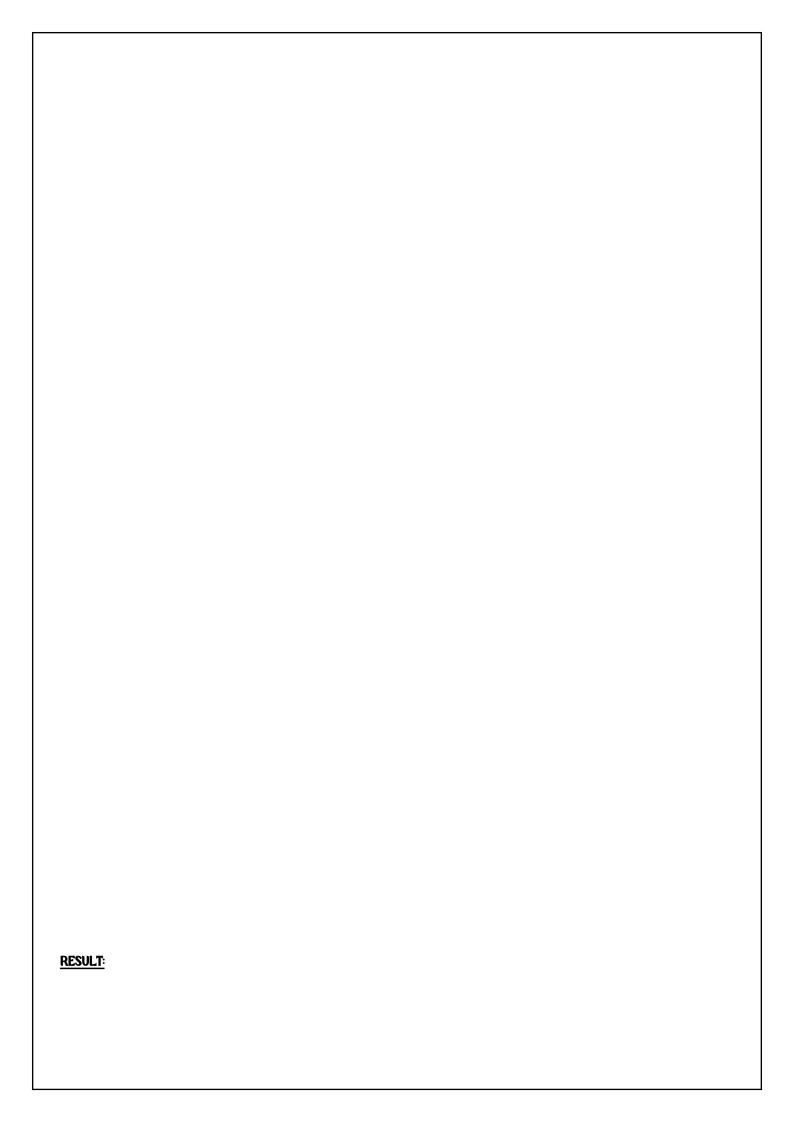
2

Output:

400

```
#include<stdio.h>
int main(){
    int m, b:
        scanf("%d", &m):
        scanf("%d", &b):
    int t=m'b:
        printf("%d", t'2):
)
```

	Input	Expected	Got	
~	100	400	400	~



EXP.NO:1(e)	DANIC C DESCRIMING DELOTION	
DATE:	BASIC C PROGRAMMING-PRACTICE	
<u>M</u> :		
<u>.GORITHM</u> :		

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.

Example Input/Output:

Input:

500

3

Output:

2100

```
#include<stdio.h>
int main()(

int a,d:

scanf("%d",&a):

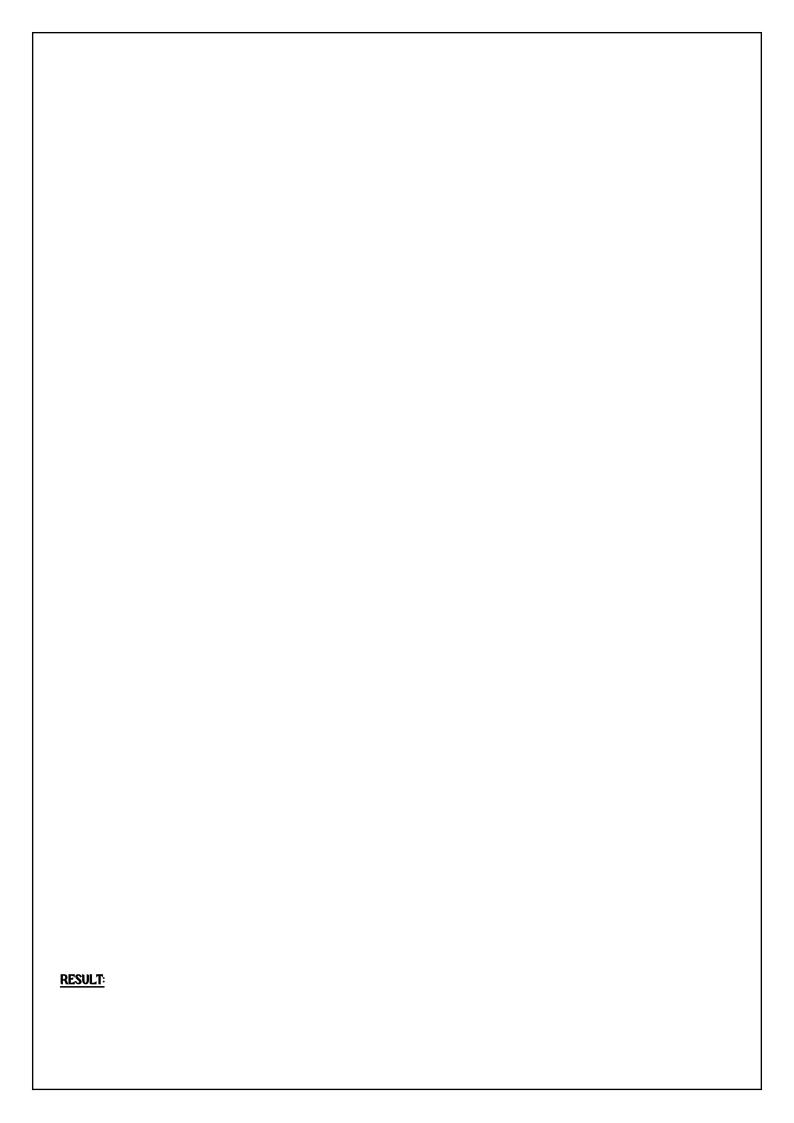
scanf("%d",&d):

int t=O:

for(int i=O:i<d:i++)
{
    a=a+200:

    t=t+a:
}
printf("%d",t-600):
}
```

	Input	Expected	Got	
~	500 3	2100	2100	*
~	100	900	900	~



XP.NO:1(f)	DACIC C DDOCDAMMING DDACTICE	
DATE:	BASIC C PROGRAMMING-PRACTICE	
<u>4:</u>		
GORITHM:		
NO TO THIS		

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 ${\bf M}$ The second line denotes the value of ${\bf N}$ The third line denotes the value of ${\bf X}$

Output Format:

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

Boundary Conditions:

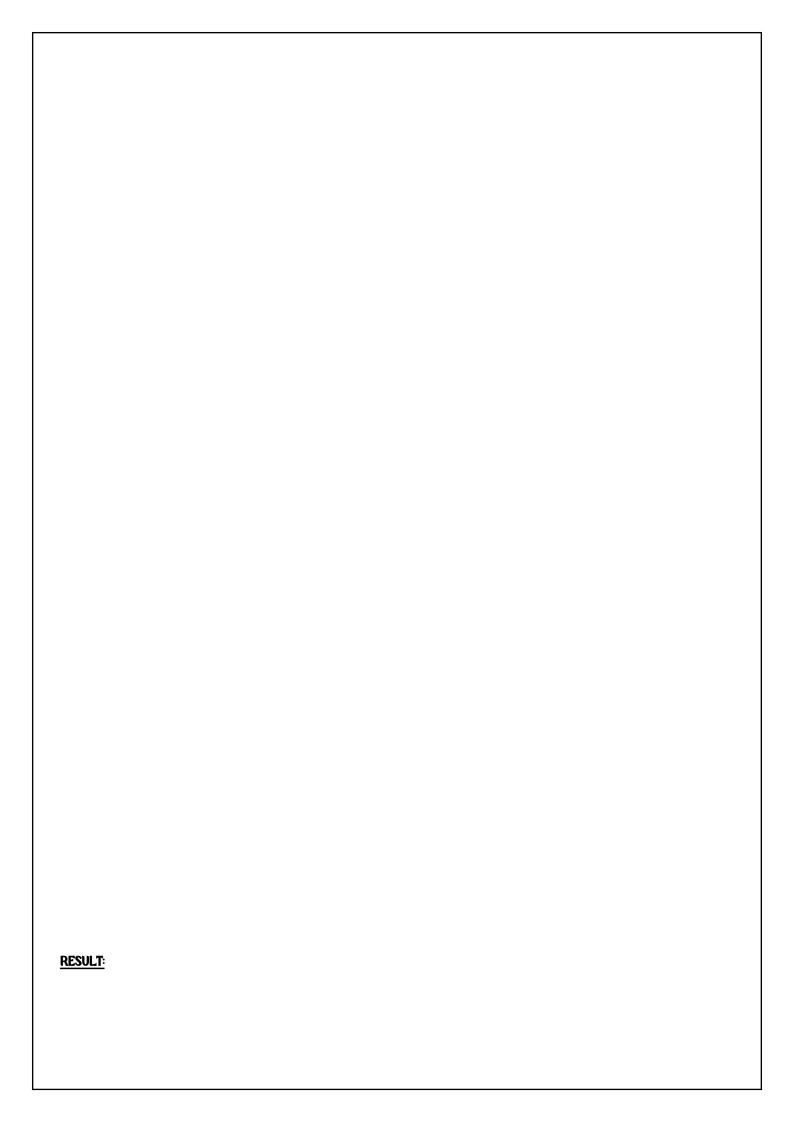
```
1 <= M <= 9999999
M < N <= 9999999
1 <= X <= 9999

Example Input/Output 1:
Input:
2
40
7
Output:
```

35 28 21 14 7

```
#include<stdio.h>
int main()
{
    int m,n,x:
        scanf("%d\n%d\n%d\n,&m,&n,&x):
    for(int i=n:i>=m:i--){
        if(i%x==0){
            printf("%d ",i):
        }
    }
}
```

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



EXP.NO:1(g)	BASIC C PROGRAMMING-PRACTICE		
DATE:	DASIC C PROGRAMMING-PRACTICE		
<u>M</u> :			
.GORITHM:			
<u>Aloriiiw</u>			

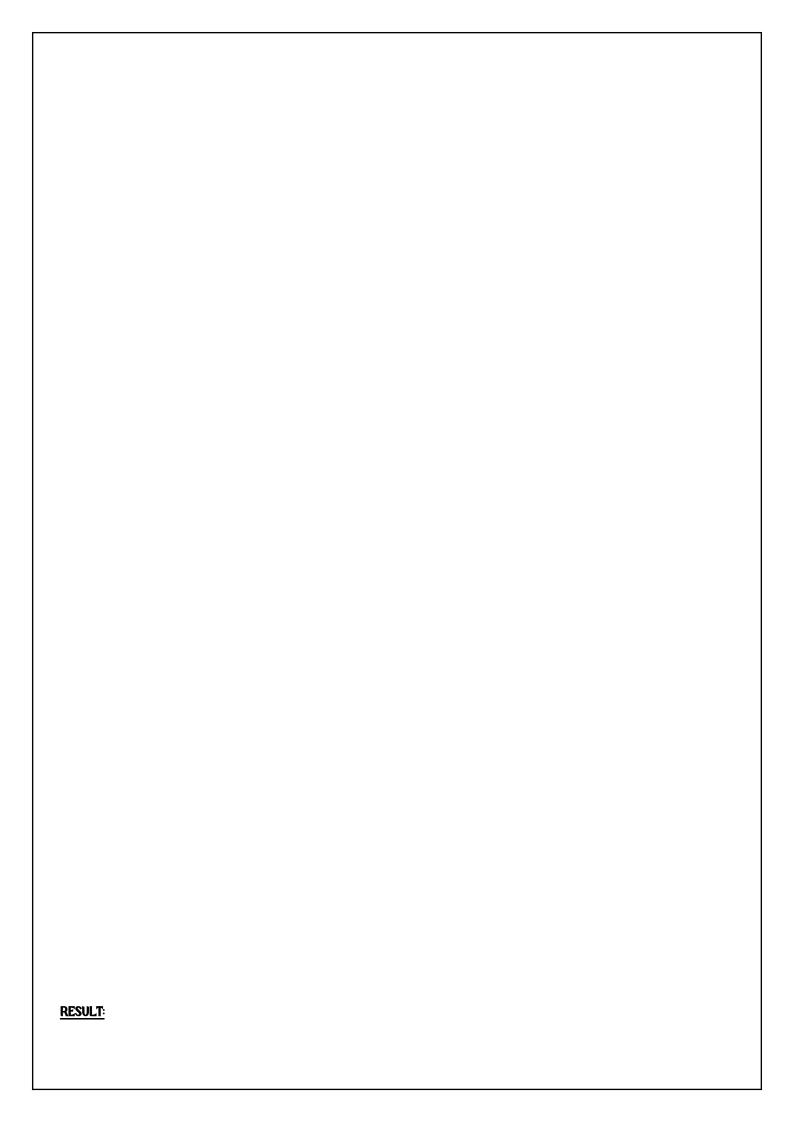
Write a C program to find the quotient and reminder of given integers.

For example:

Input	Result
12	4
3	0

```
#include<stdio.h>
int main(){
   int a, b:
    scanf("%d\n%d", & a, & b):
    printf("%d\n%d", a/b, a%b):
    return O:
}
```

	Input	Expected	Got	
~	12	4	4	~
	3	0	0	



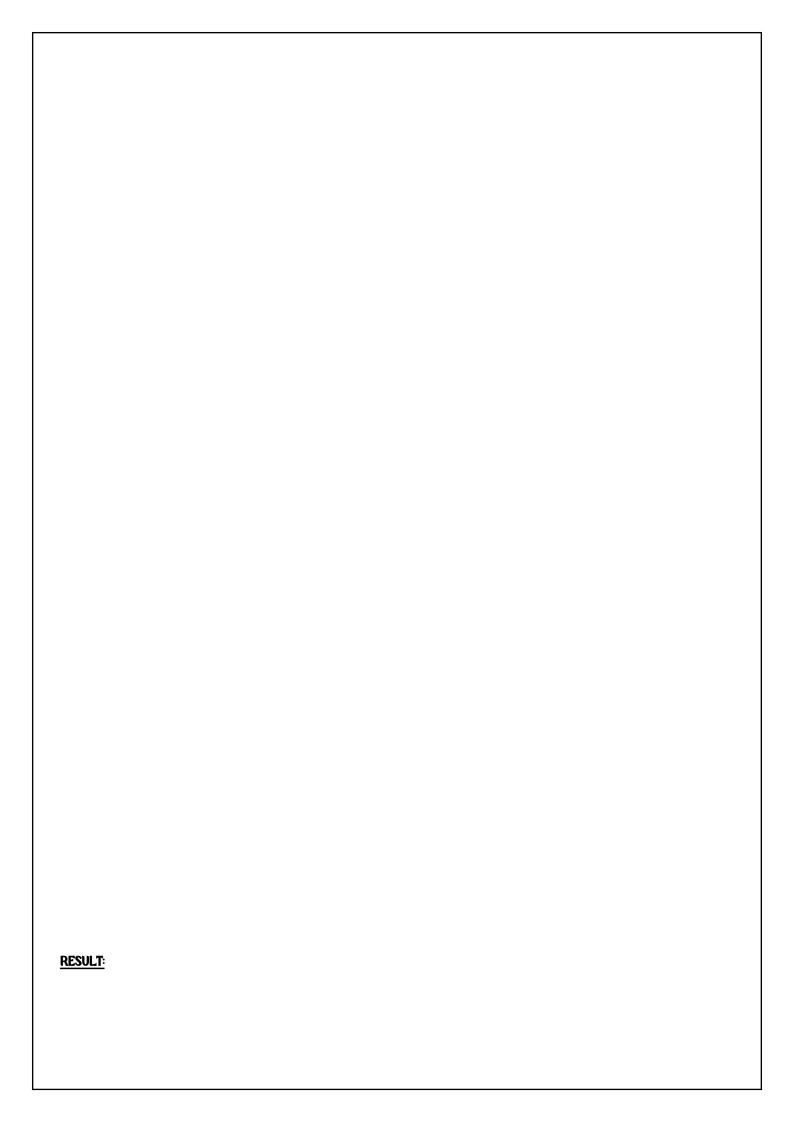
EXP.NO:1(h)			
DATE:	BASIC C PROGRAMMING-PRACTICE		
<u>M</u> :			
<u>LGORITHM</u> :			

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

For example:

In	out		Result
10	20	30	30

	Input	Expected	Got	
~	10 20 30	30	30	~



EXP.NO:1(i)	BASIC C PROGRAMMING-PRACTICE	
DATE:	DASIC C PROGRAMMING-PRACTICE	
<u>4</u> :		
GORITHM:		

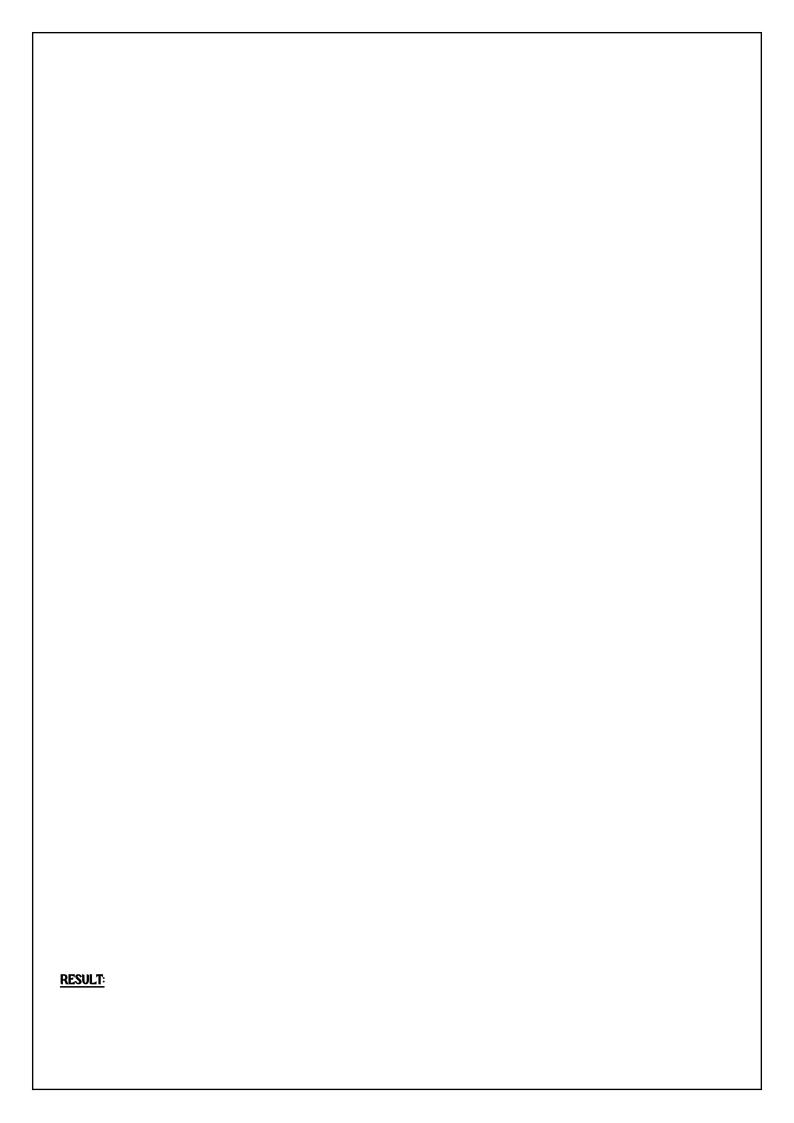
Write a C program to find whether the given integer is odd or even?

For example:

Input	Result
12	Even
11	Odd

```
#include<stdio.h>
int main()
{
   int n:
     scanf("%d",&n):
   if(n%2==0)
     printf("Even"):
   else
     printf("Odd"):
)
```

	Input	Expected	Got	
~	12	Even	Even	~
~	11	Odd	Odd	~



EXP.NO:1(j)	BASIC C PROGRAMMING-PRACTICE	
DATE:		
<u>M</u> :		
.GORITHM:		

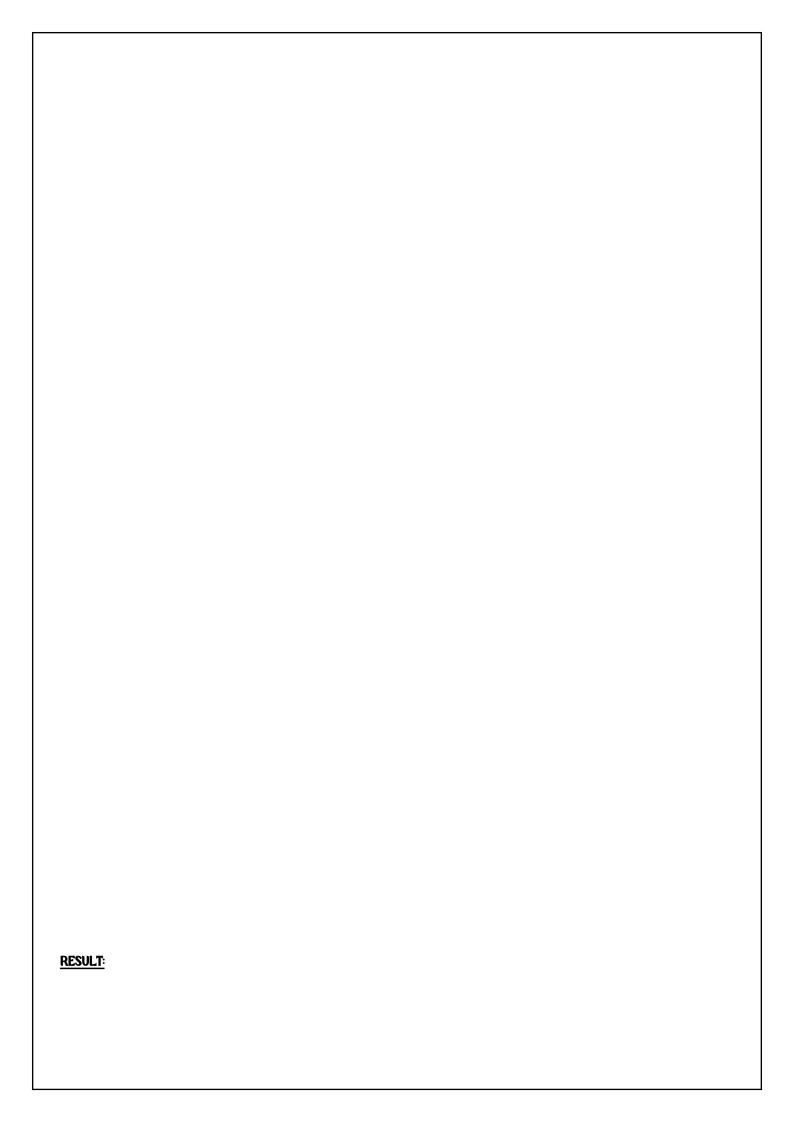
Write a C program to find the factorial of given n.

For example:

Input	Result	
5	120	

```
#include<stdio.h>
int main()
{
    int n:
        scanf(*%d*,&n):
    int i,f=1:
        for(i=1:i<=n:i++)
        (
            f'=i:
        )
        printf(*%d*,f):
}
```

	Input	Expected	Got	
~	5	120	120	~



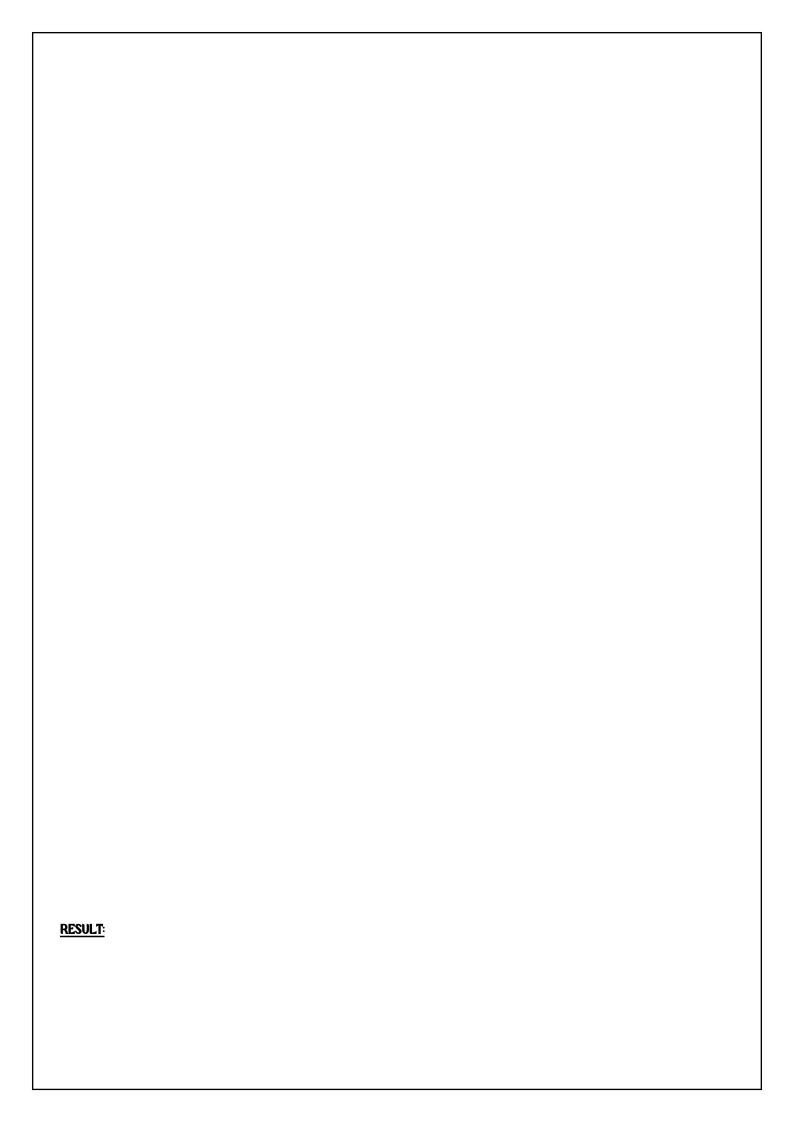
EXP.NO:1(k)	BASIC C PROGRAMMING-PRACTICE	
DATE:	DASIC C PROGRAMMING-PRACTICE	
<u>M</u> :		
<u>.gorithm:</u>		

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

For example:

Input	Result
3	6

	Input	Expected	Got	
~	3	6	6	~



EXP.NO:1(I)	PACIC C DDOCDAMMING DDACTICE	
DATE:	BASIC C PROGRAMMING-PRACTICE	
<u>M</u> :		
_GORITHM:		
<u>20001111189</u>		

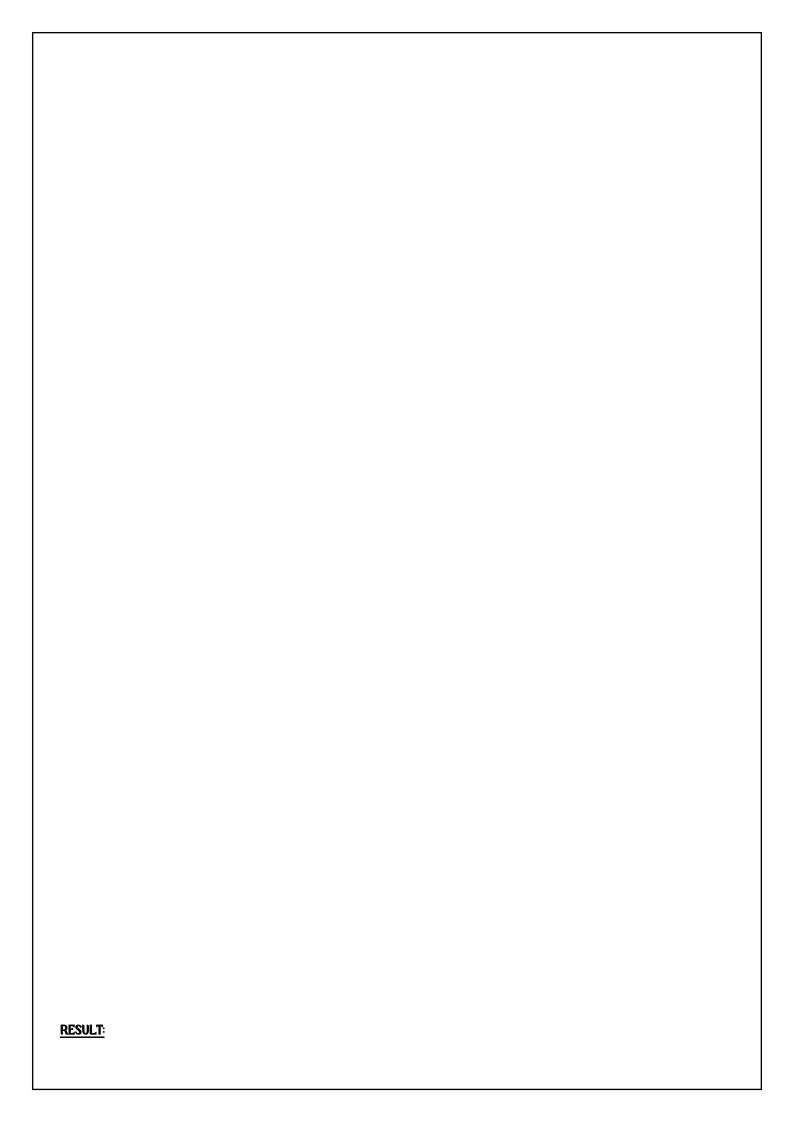
Write a C program to find the Nth term in the fibonacci series.

For example:

Input	Result
0	0
1	1
4	3

```
#include<stdio.h>
int main()(
    int n,a,b,c:
    scanf("%d",&n):
    a=0:
    b=1:
    for(int i=1:i<=n:i++)
    (
        c=a+b:
        a=b:
        b=c:
    )
    printf("%d",a):
    return 0:
)
```

	Input	Expected	Got	
~	0	0	0	~
~	1	1	1	~
~	4	3	3	~



EXP.NO:1(m)	PAGIC C DEGCENAMING DEACTICE
DATE:	BASIC C PROGRAMMING-PRACTICE
<u>M</u> :	
.GORITHM:	

Write a C program to find the power of integers.

input:

a b

output:

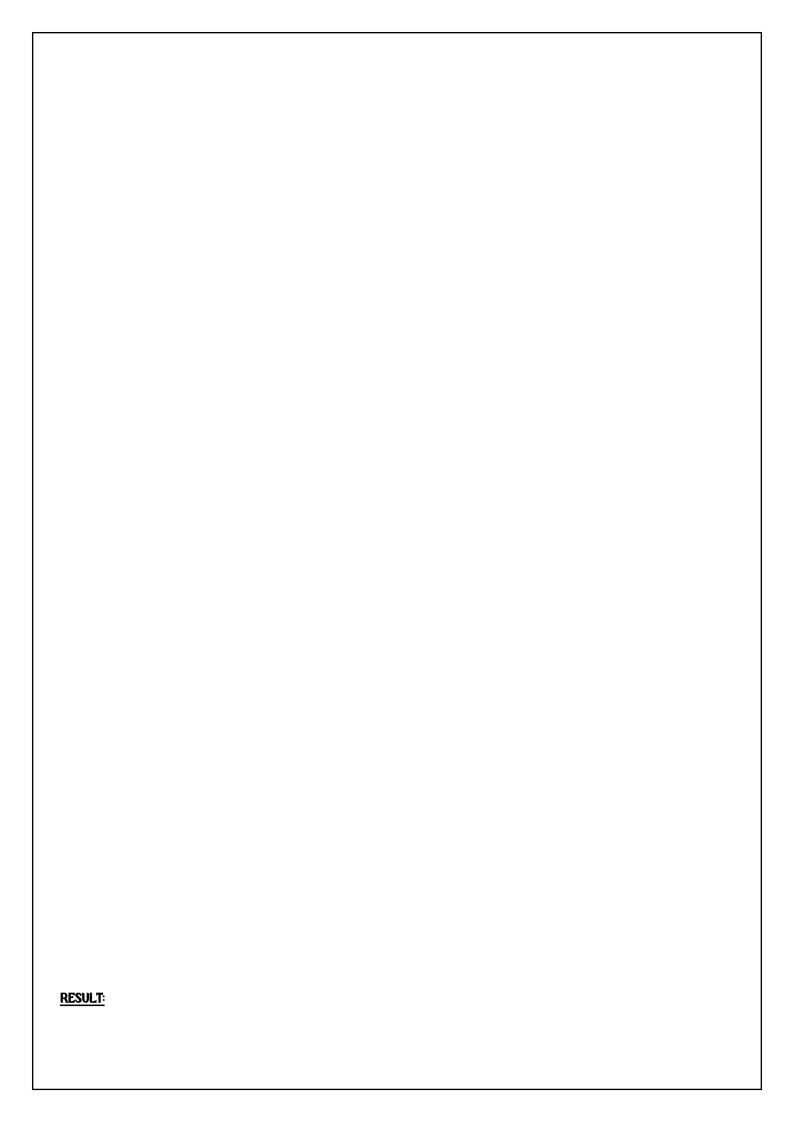
a'b value

For example:

Input	Result
2 5	32

```
#include<math.h>
#include<stdio.h>
int main()
(
    int a,b:
    scanf("%d %d",&a,&b):
    int p=pow(a,b):
    printf("%d",p):
)
```

	Input	Expected	Got	
~	2 5	32	32	~



EXP.NO:1(n)	DAOIG C DDOCDAMMING DDACTICE	
DATE:	BASIC C PROGRAMMING-PRACTICE	
<u>M</u> :		
.GORITHM:		
ZGORITINE:		

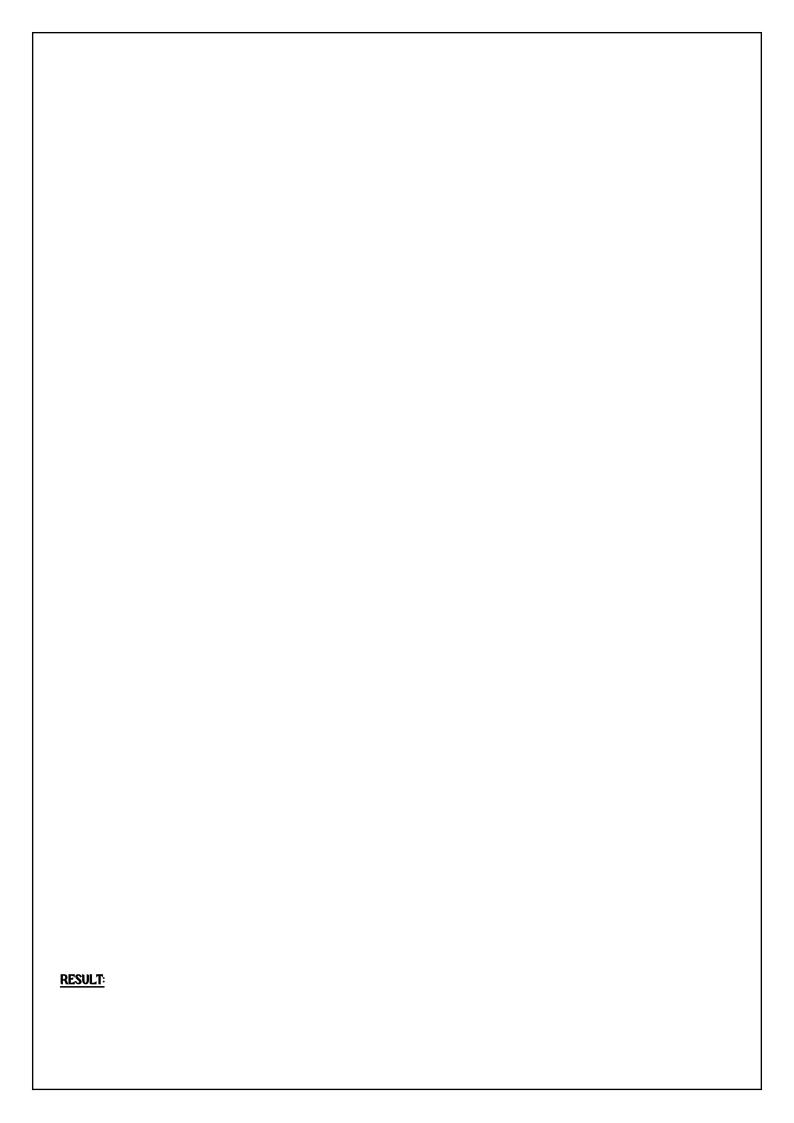
Write a C program to find Whether the given integer is prime or not.

For example:

Input	Result	
7	Prime	
9	No Prime	

```
#include<stdio.h>
int main()
{
    int n:
        scanf("%d",&n):
    int c=0:
    for(int i=1:i<=n:i++)
    {
        if(n%i==0)
        c++:
    }
    if(c==2)
    {
        printf("Prime"):
    }
    else
    {
        printf("No Prime"):
    }
```

	Input	Expected	Got	
~	7	Prime	Prime	~
~	9	No Prime	No Prime	~

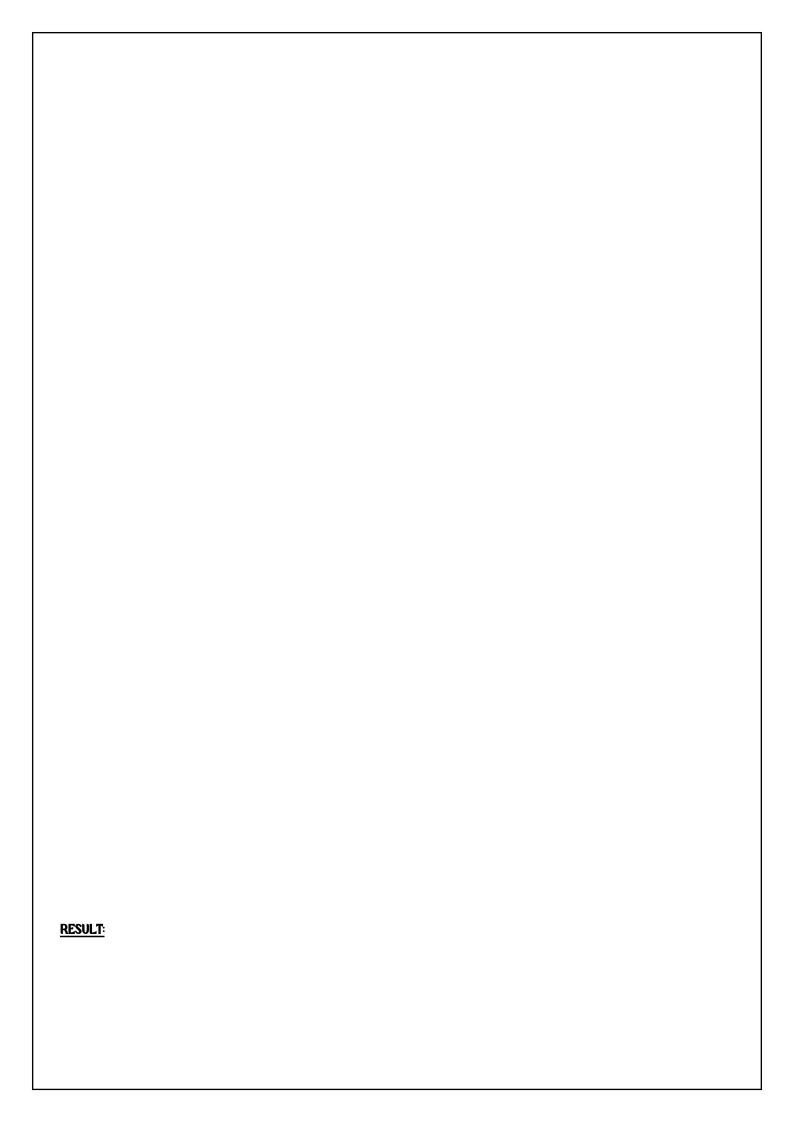


EXP.NO:1(o)	PACIC C DDOCDAMMING DDACTICE	
DATE:	BASIC C PROGRAMMING-PRACTICE	
<u>M</u> :		
<u>.GORITHM</u> :		

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

```
#include<stdio.h>
int main(){
    int n:
    scanf("%d",&n):
    int r=0:
    while(n!=0){
        r=(r'10)+(n%10):
        n=n/10:
    }
    printf("%d",r):
}
```

	Input	Expected	Got	
~	123	321	321	~



EXP.NO:2(a)	FINDING TIME COMDI EVITY DEING COUNTED METHOD
DATE:	FINDING TIME COMPLEXITY USING COUNTER METHOD
<u>M</u> :	
.GORITHM:	

```
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:
    }
}

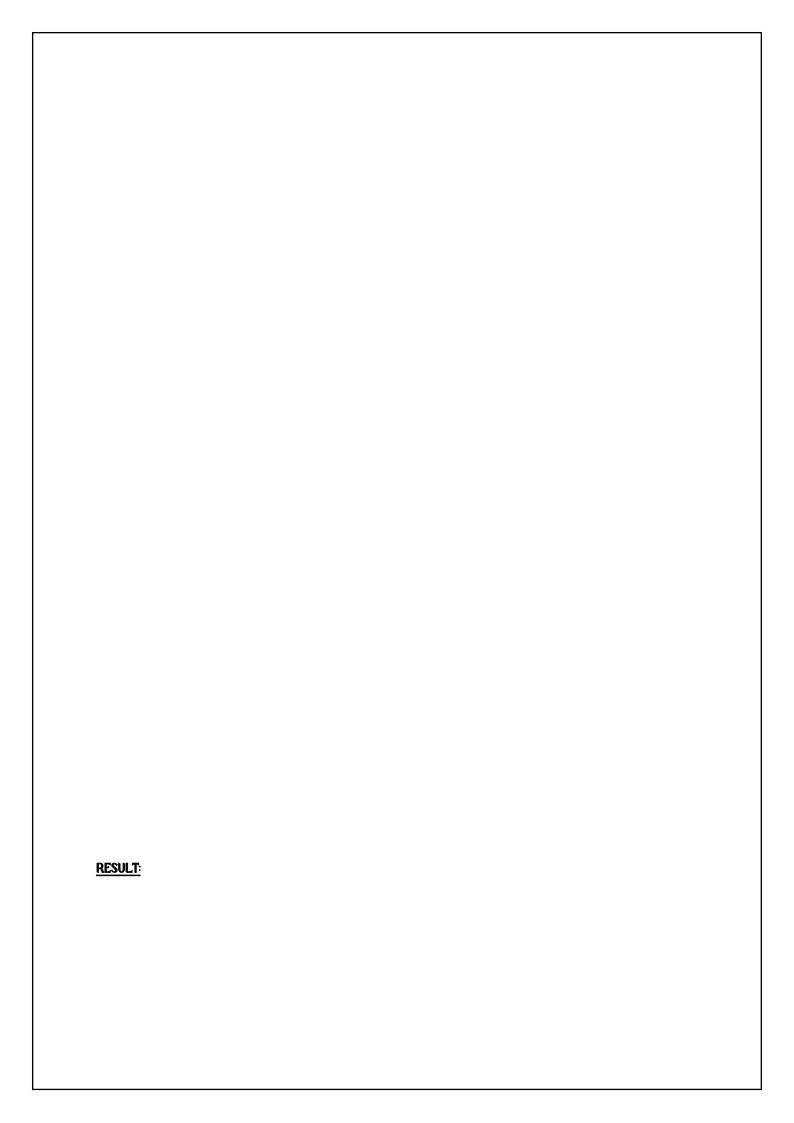
Note: No need of counter increment for declarations and scanf() and count variable printf() statements.

Input:
A positive Integer n Output:
Print the value of the counter variable
```

```
#include<stdio.h> int
main()
{
   int n:
   int count=0:
   scanf("%d",&n): int
   i=1: count++:
   int s=1: count++:
   while(s<=n)
   {
      count++:</pre>
```

```
j++;
      count++; s=s+i;
      count++;
   }
   count++;
   printf("%d",count);
   return 0;
}
```

	Input	Expected	Got	
~	9	12	12	~
~	4	9	9	~



EXP.NO:2(b)	FINDING TIME COMDI EVITY LIGING COLINTED METHOD	
DATE:	FINDING TIME COMPLEXITY USING COUNTER METHOD	
A IA.		

<u>AIM</u>:

ALGORITHM:

```
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:

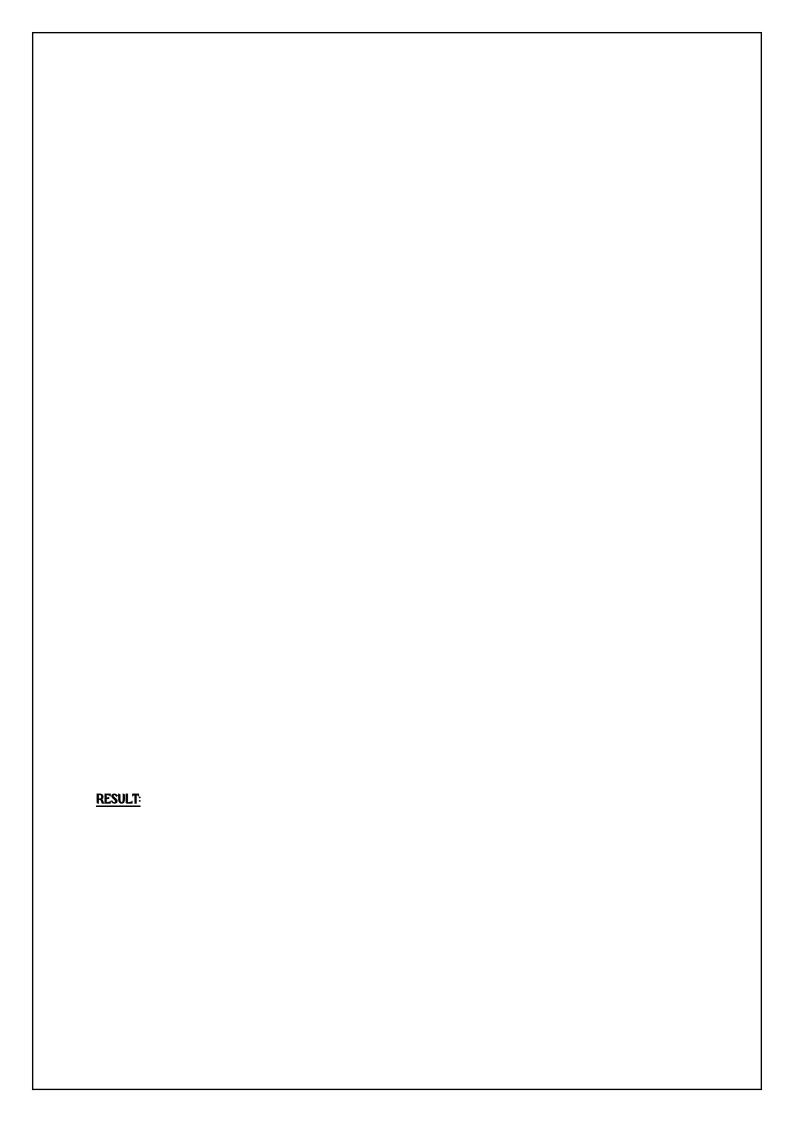
Print the value of the counter variable

PROGRAM:

}

```
#include<stdio.h> int
main()
{
    int n: scanf("%d",&n):
    int c = 0:
    int i:
        c++:
    int j:
```

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



EXP.NO:2(c)	EINDING TIME COMPLEYEY NEING COUNTED METROD	
DATE:	FINDING TIME COMPLEXITY USING COUNTER METHOD	
<u>IM:</u>		
LGORITHM:		

QUESTION:

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

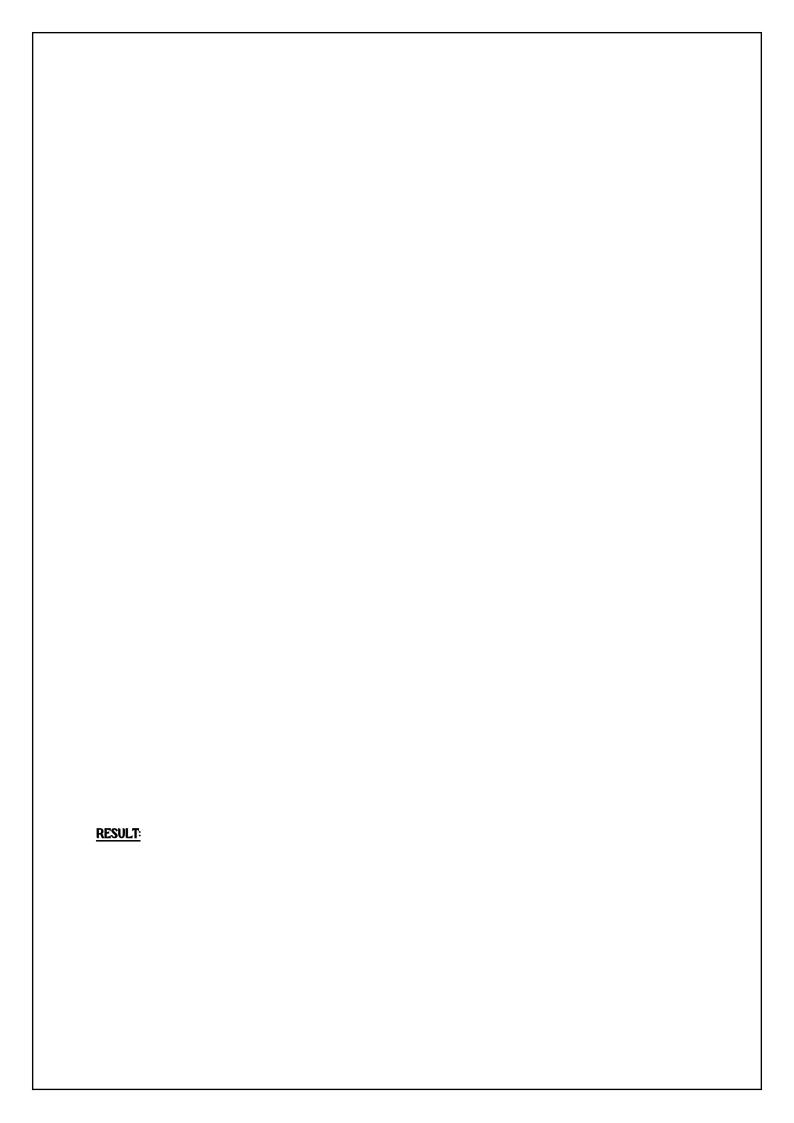
```
#include<stdio.h> int
```

```
main()
{
    int n,i: int
    c=0:
    scanf("%d",&n):

    for (i = 1: i <= n:++i)
    {
        C++:
        if (n % i== 0)
        (
            C++:
        // printf("%d *, i):
```

) c++			
) C++	•		
) c++;			
printf("%d"	,c); return		
О;			
)			

	Input	Expected	Got	
~	12	31	31	~
~	25	54	54	~
~	4	12	12	~
Passe	d all tes	tsl 🗸		



EXP.NO:2(d)	CINDING TIME COMPLEVITY HOME COUNTED METHOD	
DATE:	FINDING TIME COMPLEXITY USING COUNTER METHOD	
AIM:		
ALGOR ITHM :		

QUESTION:

using counter method.

```
Convert the following algorithm into a program and find its time complexity
```

```
void function(int n)
{
   int c= 0:
   for(int i=n/2: i<n: i++) for(int j=1: j<n:
        j = 2 ' j)
        for(int k=1: k<n: k = k ' 2) c++:
}</pre>
```

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

Input:

A positive Integer n Output:

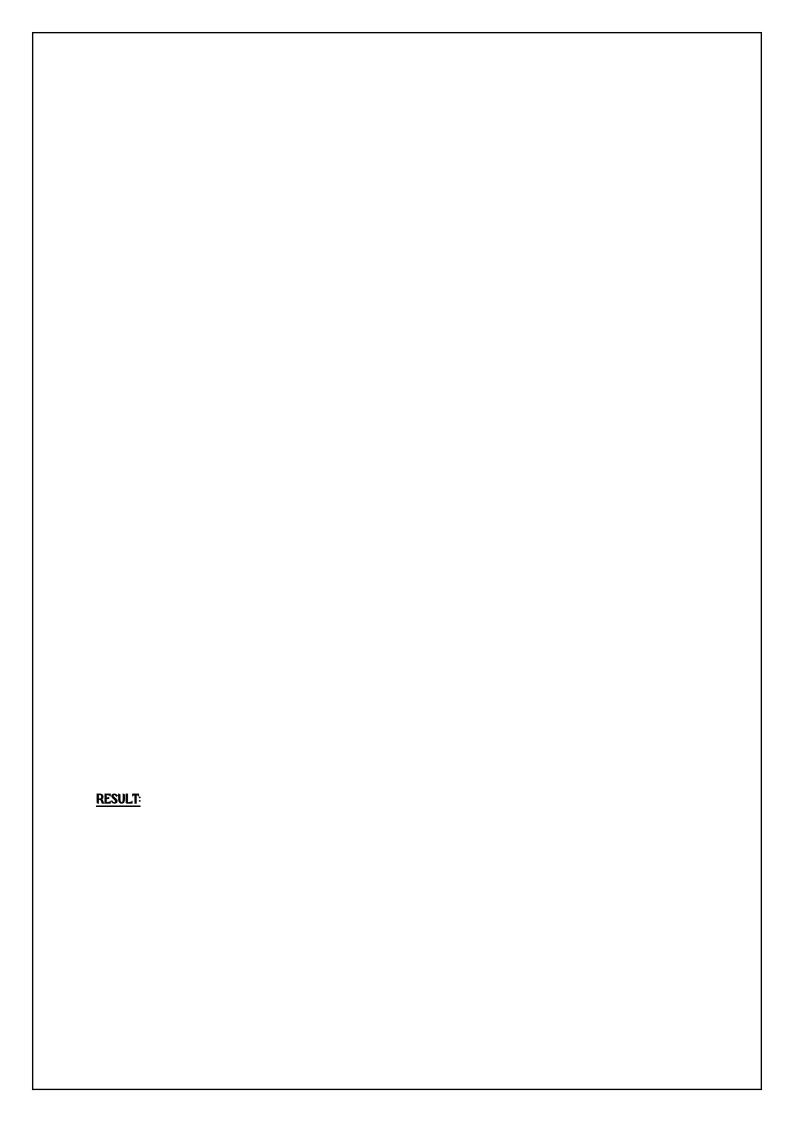
Print the value of the counter variable

```
#include<stdio.h> int
main()
{
    int n: scanf("%d",&n):
    int c=0:
        c++:
    for(int i=n/2:i<n:i++)
    {
        c++:
        for(int j =1:j<n:j=2'j)
        (
        c++:
        for(int k =1:k<n:k=k'2)
```

```
{
        C++; C++;
       // c++;
      }
      C++;
     }
   C++;
  }
  C++;
  printf("%d",c);
}
```

	Input	Expected	Got	
~	4	30	30	~
~	10	212	212	~

Passed all tests! 🗸



FINDING TIME COMPLEXITY USING COUNTER METHOD			
FINDING TIME COMPLEXITY USING COUNTER METHOD			
	FINDING TIME COMPLEXITY USING COUNTER METHOD		

ALGORITHM:

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:

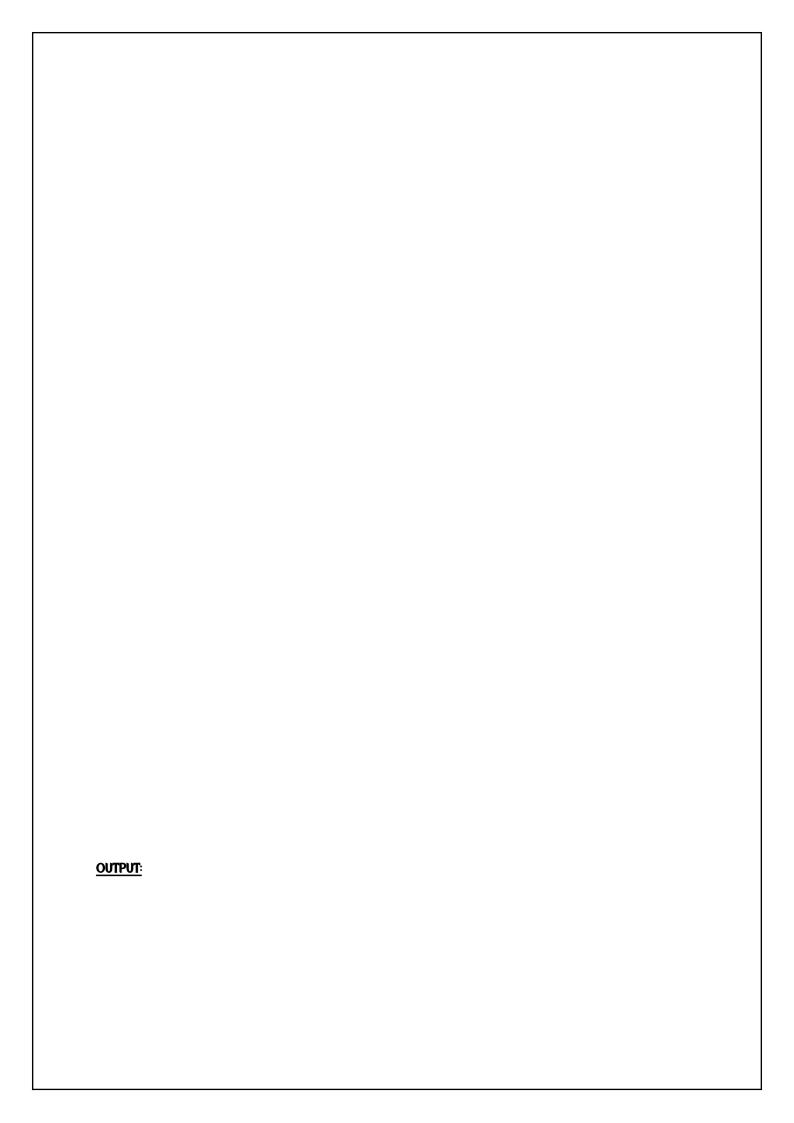
Print the value of the counter variable

```
#include<stdio.h> int
main()
{
    int n: scanf("%d",&n):
    int c =0:
    int rev =0,remainder: c++:
    while(nl=0)
    (c++:
    remainder = n % 10: c++:
```

```
rev = rev 10 + remainder; c++;
     n/= 10; c++;
}
C++;
//print(rev); c++;
printf("%d",c);
}
```

	Input	Expected	Got	
~	12	11	11	~
~	1234	19	19	~

Passed all tests! 🗸



EXP.NO:3(a)	DIVIDE AND CONQUER	
DATE:	DIVIDE AND CONQUER	
<u>IM:</u>		
LGORITHM:		

Given an array of 1s and Os this has all 1s first followed by all Os. Aim is to find the number of Os. 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

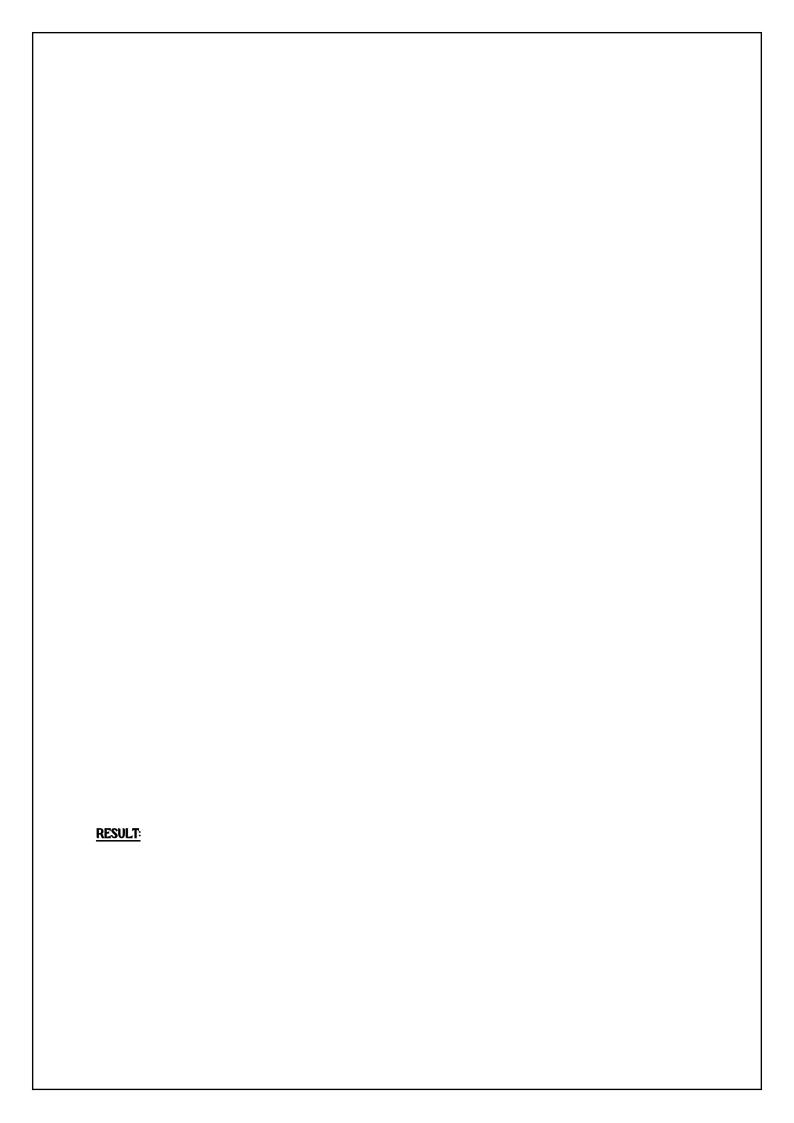
Next m lines Contains m numbers — Elements of an array Output Format

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

```
#include <stdio.h> int
main() {
   int m, i: scanf("%d", &m);
   int arr[m]:
   for(i = 0: i < m: i++) { scanf("%d",
       &arr[i]);
   }
    int low = 0, high = m - 1, mid, firstZeroIndex = -1: while(low <= high) {
       mid = low + (high - low) / 2
       if ((mid == 0 \parallel arr[mid - 1] == 1) && arr[mid] == 0) { firstZeroIndex = mid:
          break:
       if (arr[mid] == 1) { low =
          mid + 1;
       ) else {
                            high = mid - 1:
       }
   }
   if (firstZeroIndex == -1) {
```

```
printf("O\n");
   ) else {
       printf(``\%d\n``, m-firstZeroIndex);
   }
   return 0;
)
```

	Input	Expected	Got	
~	5	2	2	~
	1			
	1			
	1			
	0			
	0			
~	10	0	0	~
	1			_
	1			
	1			
	1			
	1			
	1			
	1			
	1			
	1			
	1			
~	8	8	8	~
	Ø			
	0			
	0			
	0			
	Ø			
	0			
	0			
	0			



EXP.NO:3(b) DATE:	DIVIDE A I	ND CONQUER	
<u>IM</u> :			
LGORITHM:			

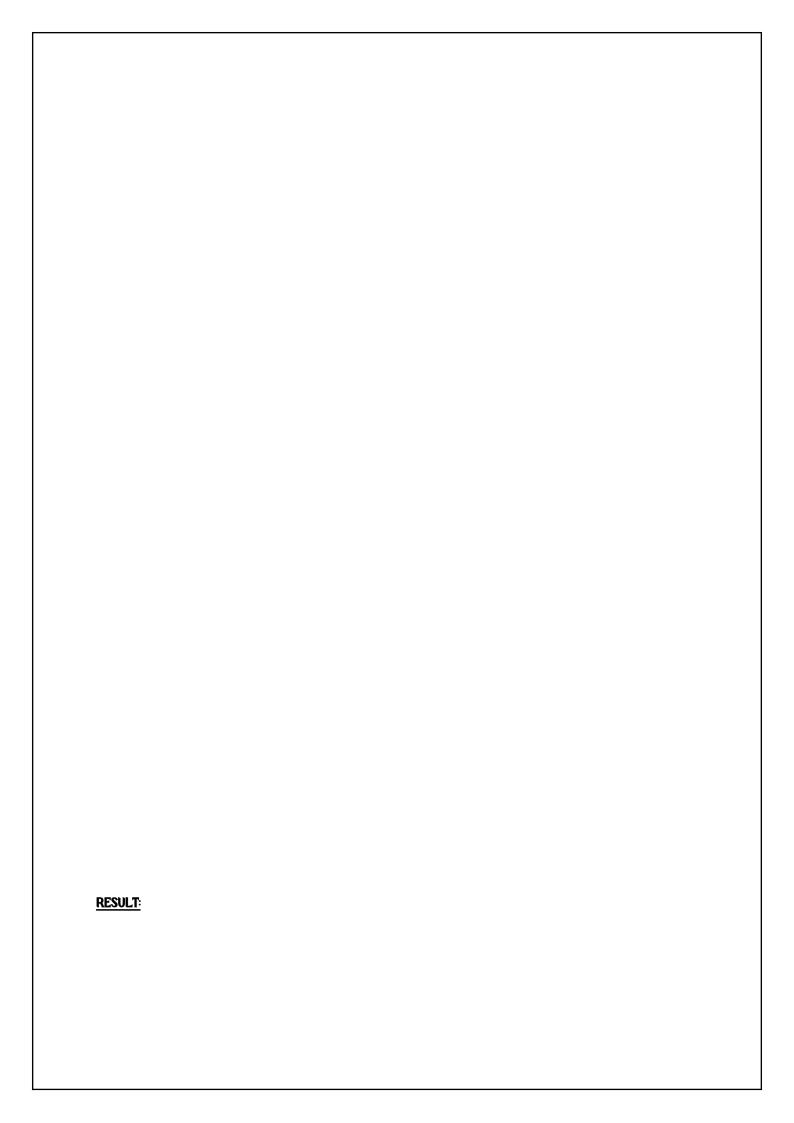
Given an array nums of size n, return the majority element.

The majority element is the element that appears more than $\ln / 2 \rfloor$ 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 · 10<sup>4</sup>
-2<sup>31</sup> <= nums[i] <= 2<sup>31</sup> - 1
PROGRAM:
#include <stdio.h>
int main() { int n;
   scanf("%d", &n); int
   nums[n];
   for (int i = 0; i < n; i++)
      { scanf("%d", &nums[i]);
   }
   int count = 0;
   int candidate = 0;
   for (int i = 0: i < n: i++) { if
      (count == 0) {
          candidate = nums[i];
      }
      if (nums[i] == candidate) { count++;
```

```
} else {
      count---:
    }
  }
  printf("%d\n", candidate);
return 0;
}
```

	Input	Expected	Got	
~	3 3 2 3	3	3	~
Passe	d all tes	ts! 🗸		



EXP.NO:3(c) DATE:	DIVIDE AND CONQUER
<u>AIM:</u>	

ALGORITHM:

```
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

PROGRAM:

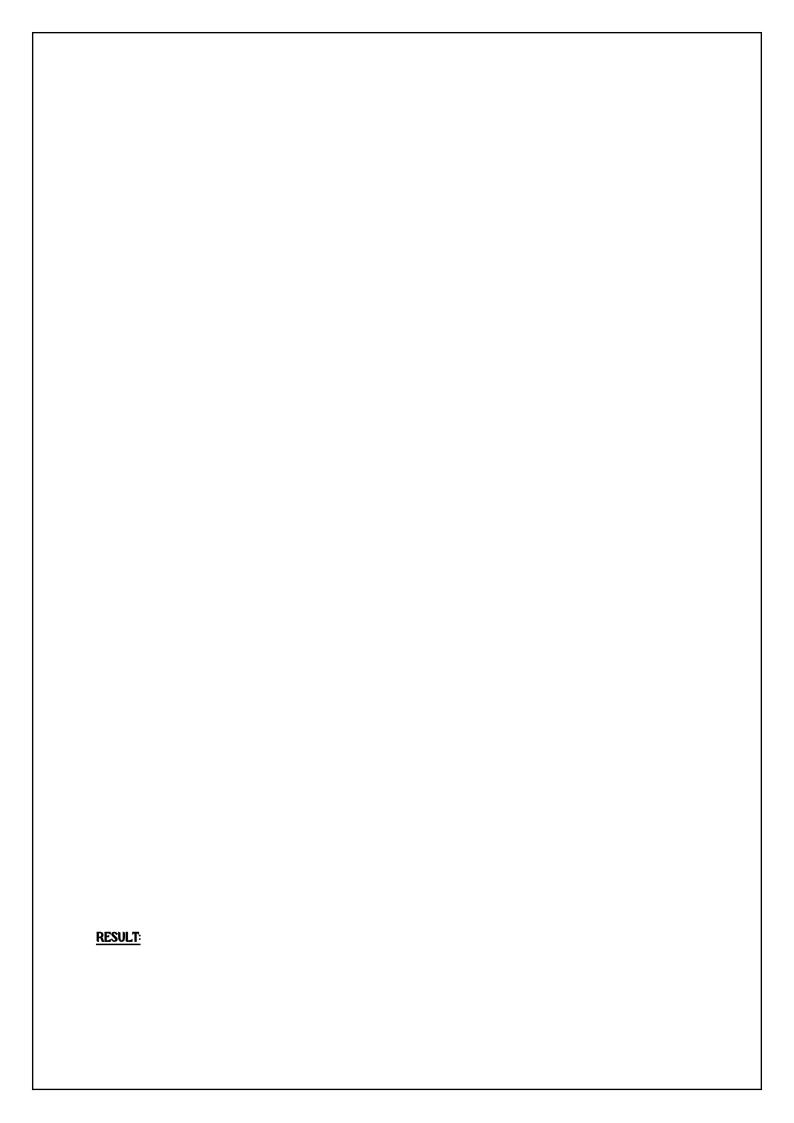
#include <stdio.h> int

main() (

int n, x: scanf("%d",
```

```
#include <stulo.n> int
main() {
    int n, x: scanf("%d",
    &n): int arr[n]:
    for (int i = 0: i < n: i++)
        { scanf("%d", & arr[i]):
    }
    scanf("%d", & x):
    int left = 0, right = n - 1: int
    floor = -1:
    while (left <= right) {
        int mid = left + (right - left) / 2: if
        (arr[mid] == x) {
            floor = arr[mid]:
                 break:
        }
        if (arr[mid] < x) {</pre>
```

	Input	Expected	Got	
~	6	2	2	~
	1			
	2			
	8			
	10			
	12			
	19			
	5			
~	5	85	85	~
	10			
	22			
	85			
	108			
	129			
	100			



EXP.NO:3(d)	
DATE:	DIVIDE AND CONQUER
AIM:	

ALGORITHM:

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

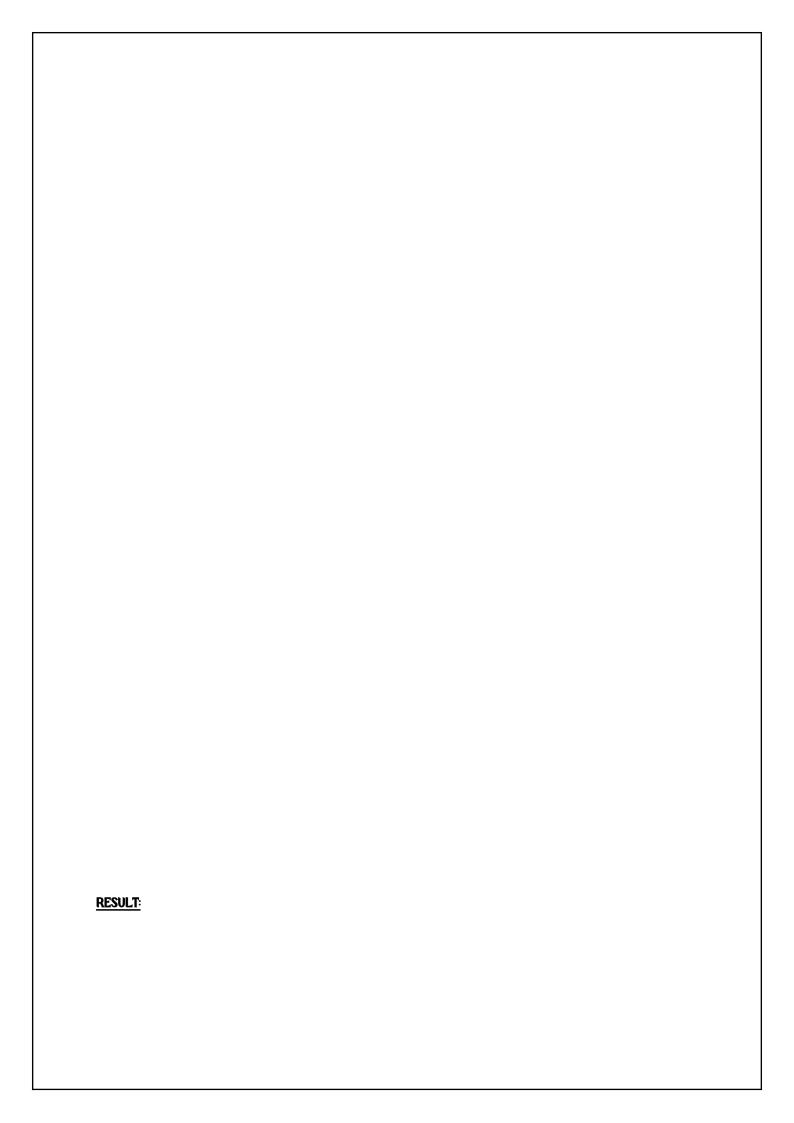
Output Format

First Line Contains Integer — Element1

Second Line Contains Integer — Element2 (Element 1 and Elements 2 together sums to value 'x')
```

```
#include <stdio.h> int
main() {
    int n, x: scanf('%d',
    &n): int arr[n]:
    for (int i = 0: i < n: i++)
        ( scanf('%d', & arr[i]):
    }
    scanf('%d', &x):
    int left = 0, right = n - 1: int
    found = 0:
    while (left < right) {
        int sum = arr[left] + arr[right]: if
        (sum == x) {
            printf('%d\n', arr[left]):
            printf('%d\n', arr[right]):
```

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! 🗸						
	4 2 4 8 10 14 5 2 4 6 8 10 10 10	4 4 2 10 4 8 10 14 5 No 2 4 6 8 10 100	2 10 10 4 8 10 14 5 No No No 2 4 6 8 10 100			



EXP.NO:3(e)	DIVIDE AND CONQUER	
DATE:		

<u>AIM</u>:

PROBLEM STATEMENT:

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:

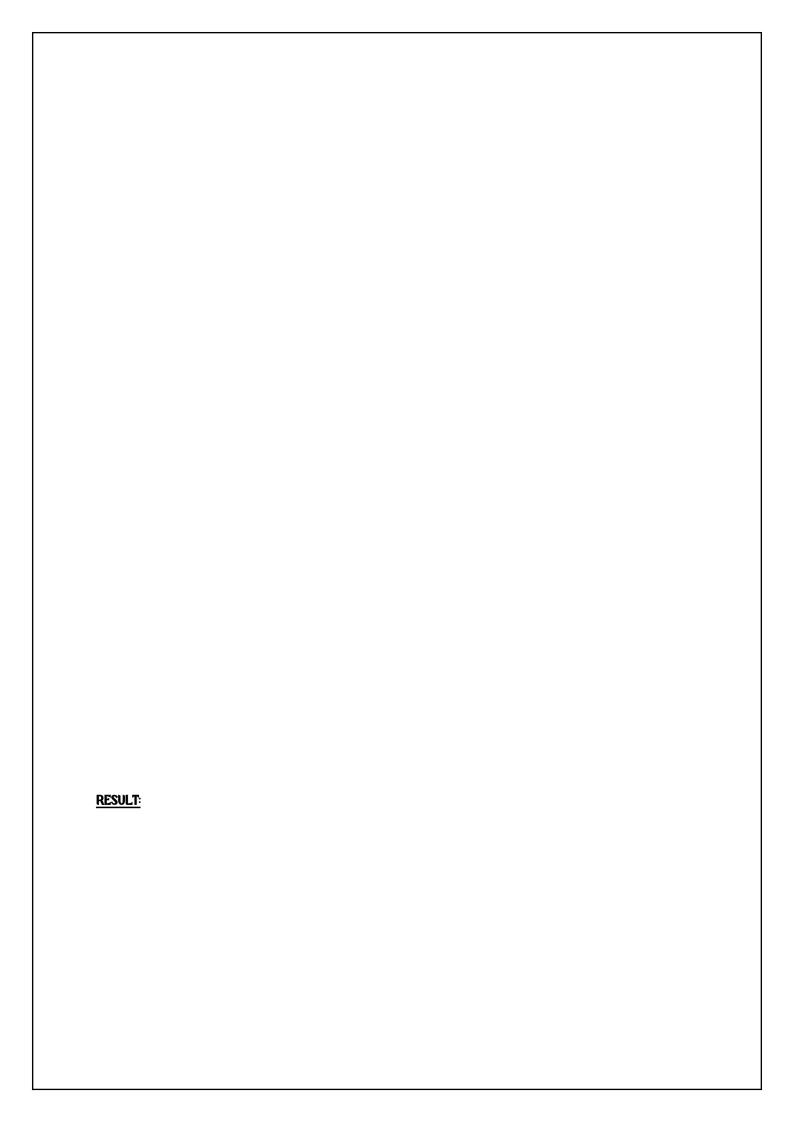
Sorted list of elements

```
#include <stdio.h> int
main() {
   int n; scanf("%d", &n);
   int a[n]:
   for (int i = 0; i < n; i++)
      { scanf("%d", &a[i]);
   }
   for (int i = 0; i < n; i++) {
      for (int j = i + 1; j < n; j++) { if (a[j] <
          a[i]) {
             int temp = a[i]; a[i] =
             a[j];
             a[j] = temp;
          }
      }
   }
   for (int i = 0; i < n; i++) {
```

```
printf("%d ", a[i]):
   }
   return 0;
}
```

	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 90 114	~
~	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! 🗸



DATE: AIM:	_			1_C_COIN DDORLE	EXP.NO:4(a)	
<u>AIM</u> :		1-G-COIN PROBLEM		DATE:		
					<u>IM:</u>	

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:

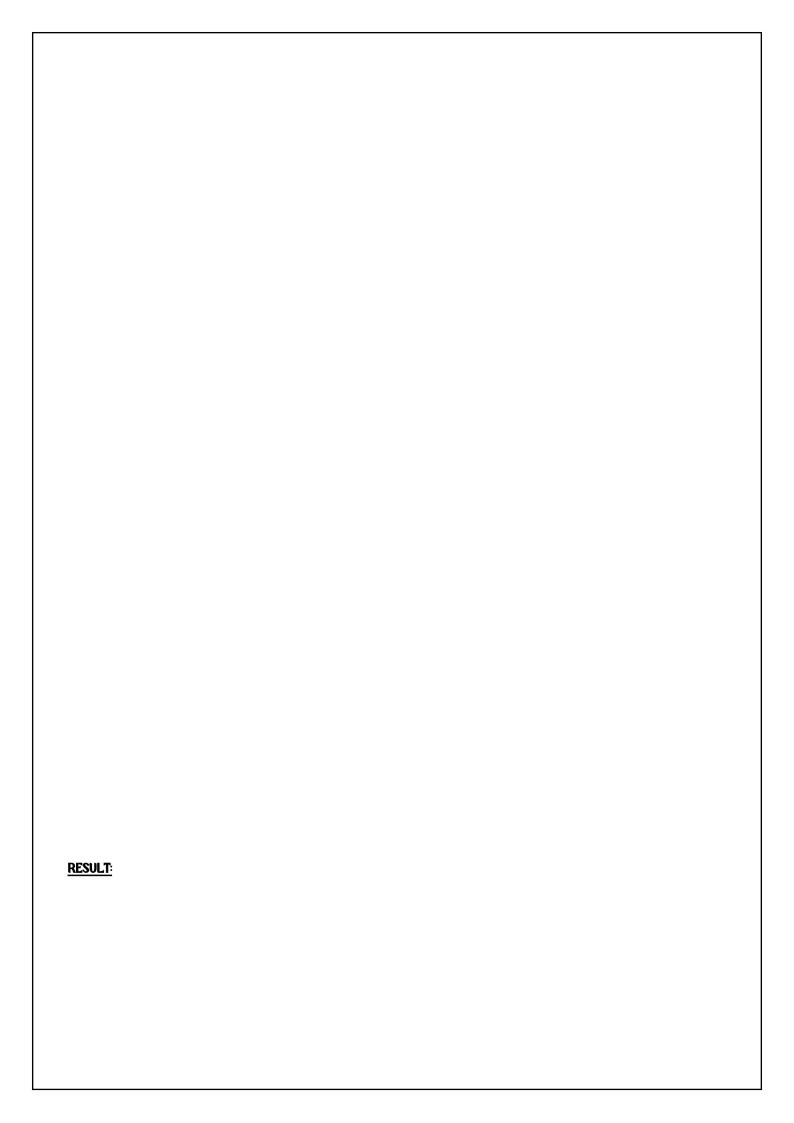
We need a 50 Rs note and a 10 Rs note and two 2 rupee coins.

PROGRAM:

Explanation:

```
return count:
}
int main() { int V;
   scanf("%d", &V);
   printf(*%d\n', min_coins_and_notes(V)): return 0:
}
```

	Input	Expected	Got	
~	49	5	5	~



EXP.NO:4(b)	2-G-COOKIES PROBLEM
DATE:	

AIM:

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] >= 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

123

2

11

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 <= g.length <= 3 104
```

0 <= s.length <= 3 104

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

PROGRAM:

#include <stdio.h> #include

<stdlib.h>

int compare(const void a, const void b) { return ((int)a

- (int ')b);

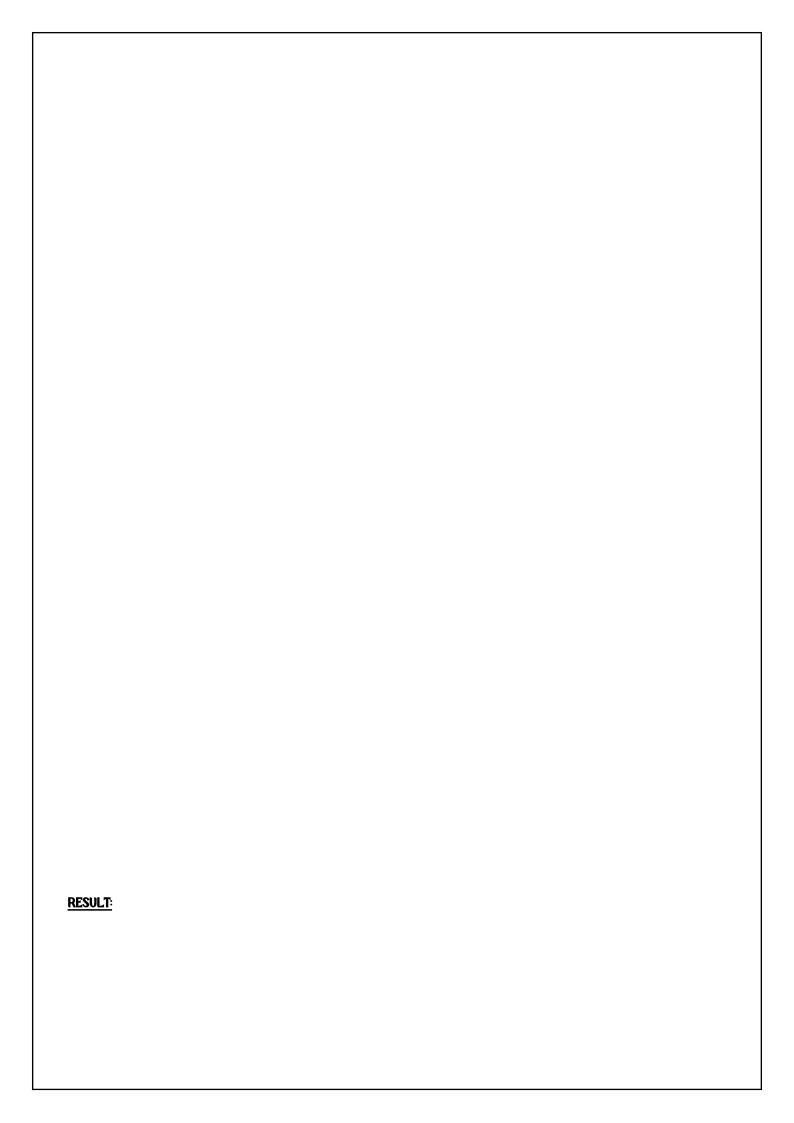
```
}
int findContentChildren(int o[], int oSize, int s[], int sSize) { qsort(o,
   gSize, sizeof(int), compare):
   qsort(s, sSize, sizeof(int), compare); int
   childIndex = 0;
   int cookielndex = 0;
   while (childlndex < gSize && cookielndex < sSize) { if
      (s[cookieIndex] >= g[childIndex]) {
         childIndex++;
      }
      cookieIndex++;
   return childlndex:
}
int main() {
   int gSize, sSize;
   scanf("%d", &gSize);
   int 'g = (int ')malloc(gSize 'sizeof(int)): if (g ==
   NULL) {
      fprintf(stderr, "Memory allocation failed\n"); return 1;
   }
   for (int i = 0: i < gSize: i++) { scanf("%d", 
      &g[i]);
   scanf("%d", &sSize);
   int's = (int')malloc(sSize 'sizeof(int)): if (s ==
   NULL) {
```

```
fprintf(stderr, 'Memory allocation failed\n'): free(q):
    return 1:
}

for (int i = 0: i < sSize: i++) { scanf('%d',
    &s[i]):
}

printf('%d\n', findContentChildren(q, qSize, s, sSize)): free(q):
    free(s):
    return 0:
}</pre>
```

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



EXP.NO:4(c) DATE:	3-G-BURGER PROBLEM
AIM:	

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 $3^i \cdot c$ 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 $(3^{0} \cdot 1) + (3^{1} \cdot 3) + (3^{2} \cdot 2) = 1 + 9 + 18 = 28$.

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

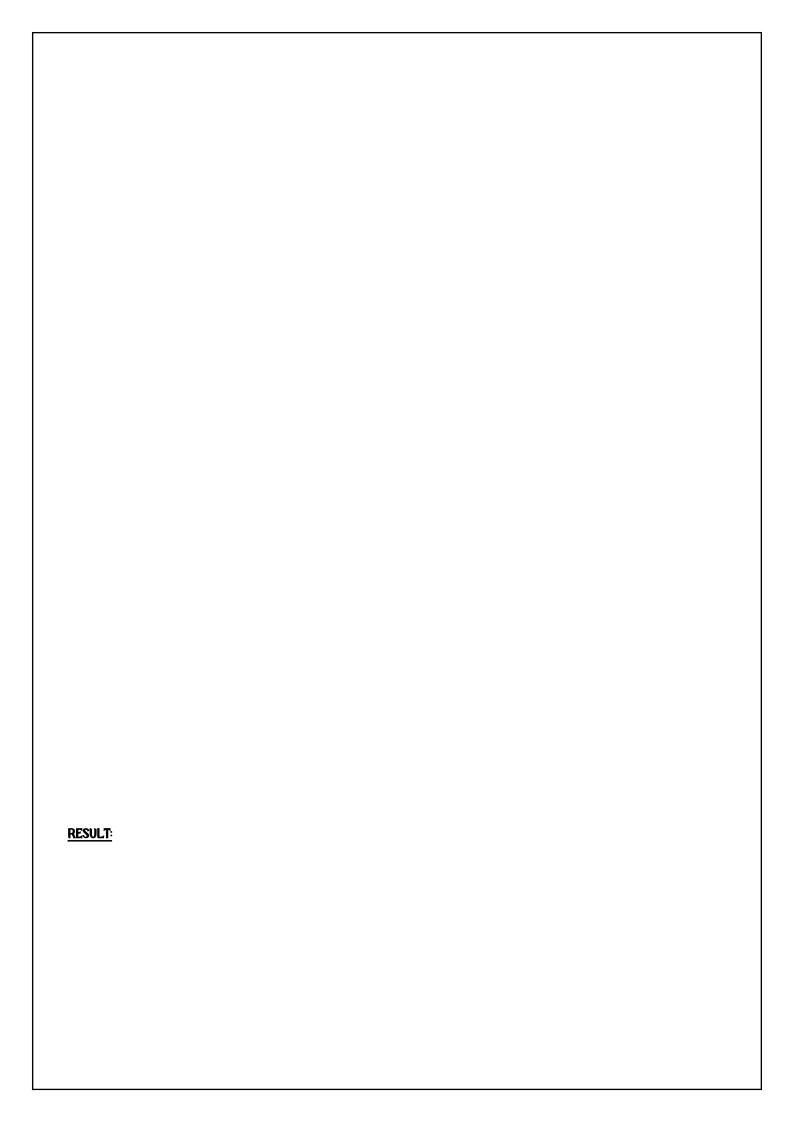
```
long long minDistance(int calories[], int n) { qsort(calories, n,
   sizeof(int), compareDescending): long long totalDistance = 0:
   long long powerOf3 = 1: for
   (int i = 0; i < n; i++) (
      if (powerOf3 > LLONG_MAX / calories[i]) {
         fprintf(stderr, 'Integer overflow detected during calculation.\n'); exit(1);
      }
      totalDistance += powerOf3 calories[i]; if
      (powerOf3 > LLONG_MAX / 3) {
         fprintf(stderr, 'Integer overflow detected while computing powers of 3.\n'); exit(1);
      }
      powerOf3 = 3;
   }
   return totalDistance:
}
int main() { int n;
   if (scanf(\%d\%, \&n) = 1 || n < 0) {
      fprintf(stderr, "Invalid input for number of burgers.\n"); return 1;
   }
   if (n == 0)
      { printf("O\n");
      return 0:
   }
```

```
int 'calories = (int ')malloc(n' sizeof(int)): if (calories
== NULL) {
    fprintf(stderr, 'Memory allocation failed\n'): return 1:
}

for (int i = 0: i < n: i++) {
    if (scanf('%d', &calories[i]) != 1 || calories[i] < 0)
        ( fprintf(stderr, 'Invalid input for calorie count.\n'):
        free(calories):
        return 1:
        )
}

printf('%lld\n', minDistance(calories, n)): free(calories):
    return 0:
}</pre>
```

	Test	Input	Expected	Got	
~	Test Case 1	3 1 3 2	18	18	~
~	Test Case 3	3 5 10 7	76	76	~



EXP.NO:4(d)	4-G-ARRAY SUM MAX PROBLEM
DATE:	

AIM:

Given an array of N integer, we have to maximize the sum of arr[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

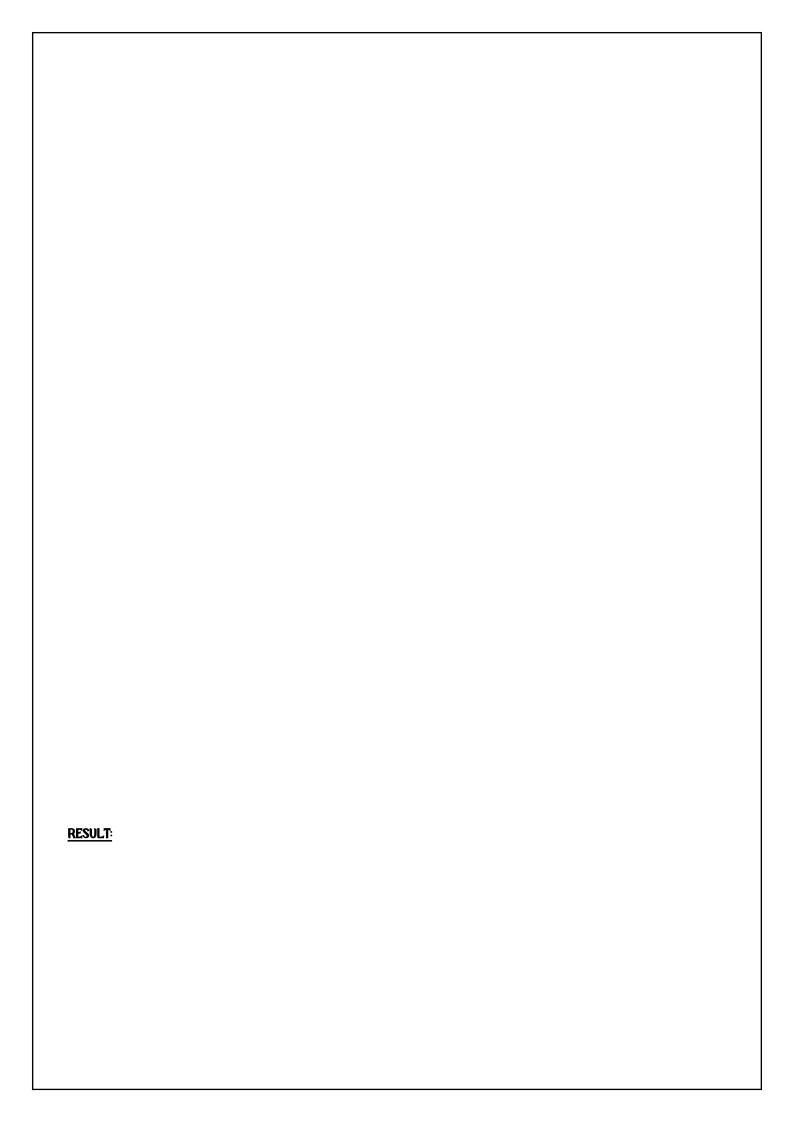
25340

Sample output:

40

```
int max_sum = 0:
   for (int i = 0; i < n; i++)
      { max_sum += arr[i] i;
   }
   printf("%d\n", max_sum);
   free(arr);
   return 0;
}
```

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



EXP.NO:4(e)	5-G-PRODUCT OF ARRAY ELEMENTS-MINIMUM
DATE:	

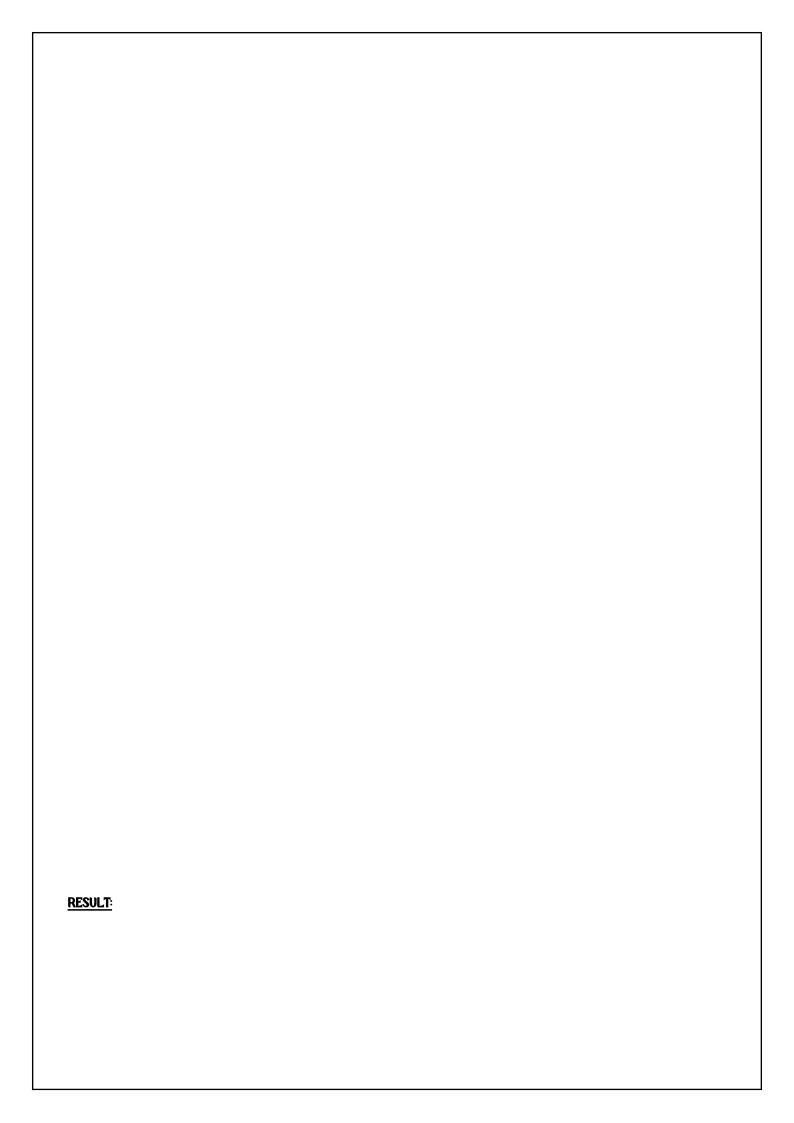
AIM:

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 SUM (A[i] \dot{B} B[i]) for all i is minimum.

```
#include <stdio.h> #include
<stdlib.h>
int compare_asc(const void a, const void b) { return
   ('(int')a - '(int')b);
}
int compare_desc(const void a const void b) { return ((int)b -
   (int )a);
}
int main() { int n;
   scanf("%d", &n);
   int array_One = malloc(n sizeof(int)); int
   array_Two = malloc(n sizeof(int)); for (int i = 0;
   i < n; i++) {
      scanf("%d", &array_One[i]);
   }
   for (int i = 0; i < n; i++) { scanf("%d",
      &array_Two[i]);
   }
   qsort(array_One, n, sizeof(int), compare_asc); qsort(array_Two, n,
   sizeof(int), compare_desc);
```

```
int min_sum = 0:
for (int i = 0: i < n: i++) {
    min_sum += array_One[i] ` array_Two[i]:
}
printf("%d\n", min_sum):
free(array_One):
free(array_Two):
    return 0:
}</pre>
```

	Input	Expected	Got	
~	3 1 2 3 4 5 6	28	28	*
•	4 7 5 1 2 1 3 4	22	22	*
•	5 20 10 30 10 40 8 9 4 3 10	590	590	~



EXP.NO:5(a)	PLAYING WITH NUMBERS	
DATE:	PLATING WITH NUMBERS	
<u>IM:</u>		
LGORITHM:		

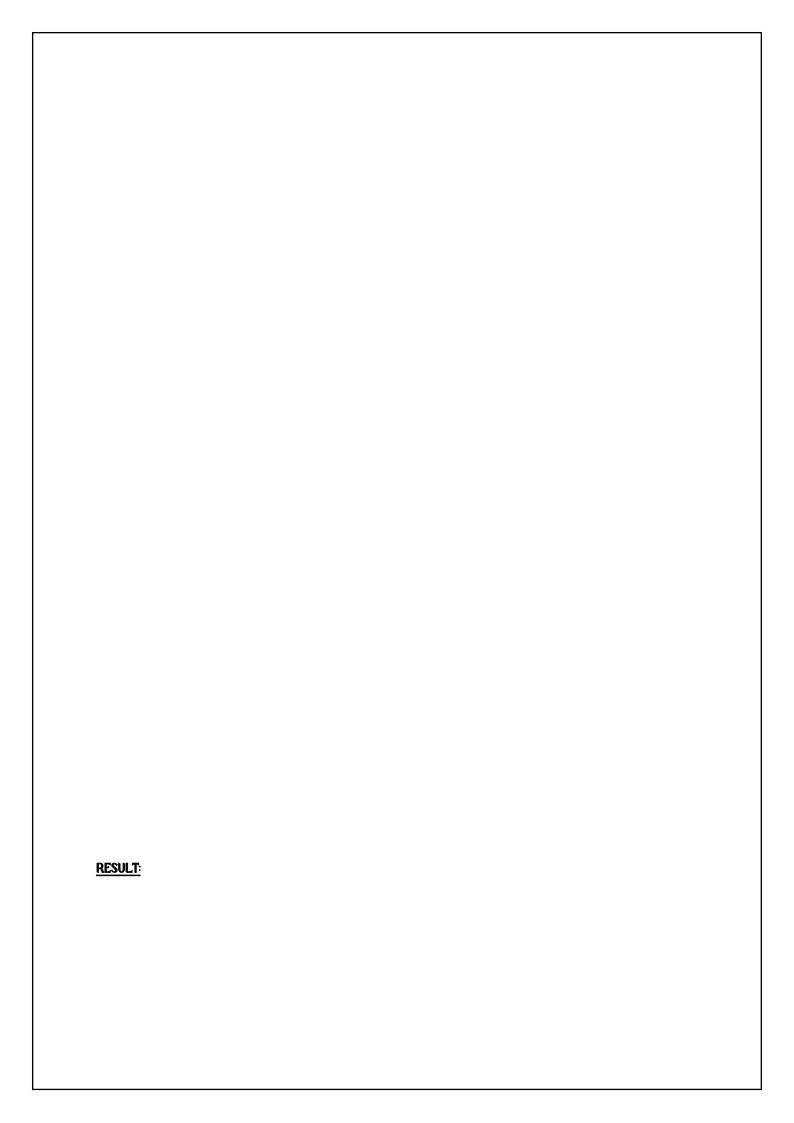
}

if $(n \ge 2)$ {

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 in can be represented using 1 and 3 Sample Input
6
Sample Output 6
PROGRAM:
#include <stdio.h>
long long count_ways(int n) { long
   long dp[n + 1];
   dp[O] = 1
       if (n >= 1) {
          dp[1] = 1;
```

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



EXP.NO:5(b)	PLAYING WITH CHESSBOARD
DATE:	

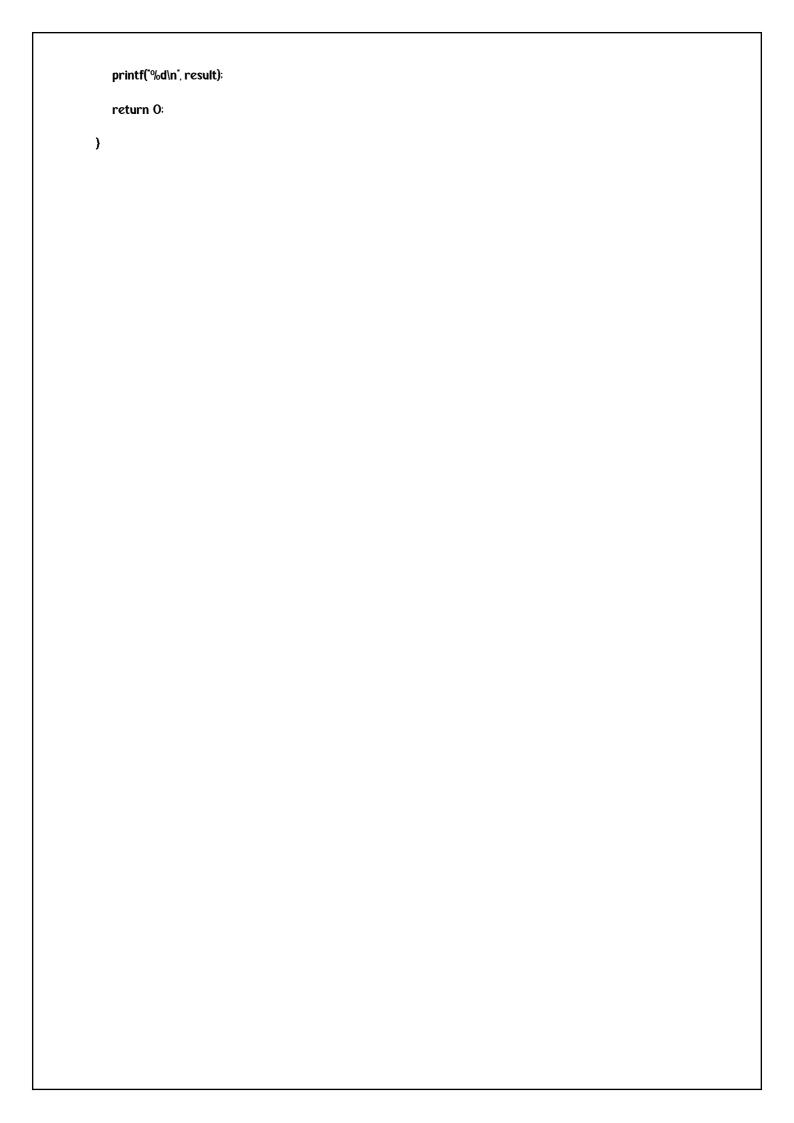
<u>AIM</u>:

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.

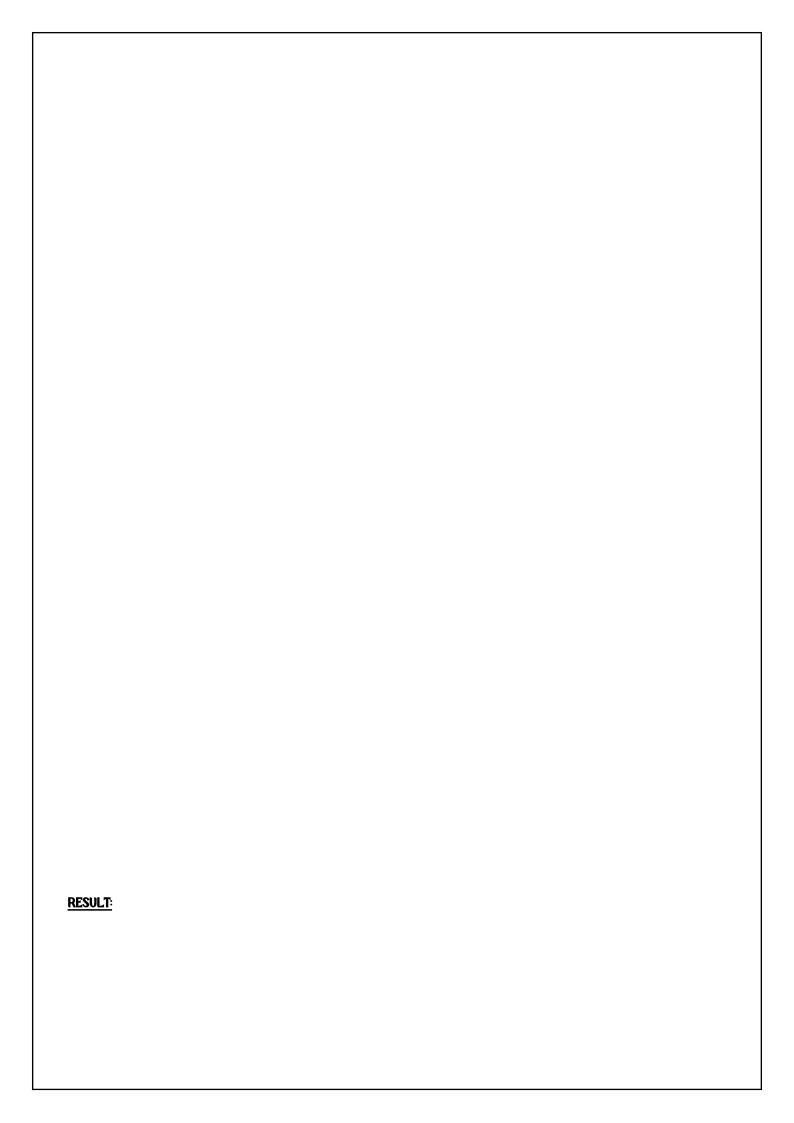
```
Example:
Input
3
1 2 4
2 3 4
8 7 1
Output:
19
Explanation:
Totally there will be 6 paths among that the optimal is Optimal path value:1+2+8+7+1=19
Input Format
First Line contains the integer n
The next n lines contain the n'n chessboard values Output
Format
```

Print Maximum monetary value of the path

```
dp[i][j] = 0;
      }
   }
   dp[O][O] = chessboard[O][O]; for
   (int j = 1; j < n; j++) {
      dp[O][j] = dp[O][j - 1] + chessboard[O][j];
   }
   for (int i = 1; i < n; i++) {
      dp[i][O] = dp[i - 1][O] + chessboard[i][O];
   }
   for (int i = 1: i < n: i++) { for (int j =
      1; j < n; j++) {
          dp[i][j] = chessboard[i][j] + max(dp[i - 1][j], dp[i][j - 1]);
      }
   }
   return dp[n - 1][n - 1];
}
int main() { int n:
   int chessboard[MAX][MAX];
   scanf("%d", &n);
   for (int i = 0: i < n: i++) { for (int j
      = 0; j < n; j++) {
          scanf("%d", & chessboard[i][j]);
      }
   }
   int result = maxMonetaryPath(chessboard, n);
```



	Input	Expected	Got	
~	3	19	19	~
	1 2 4			
	2 3 4			
	8 7 1			
~	3	12	12	~
	1 3 1			
	151			
	4 2 1			
~	4	28	28	~
	1 1 3 4			
	1 5 7 8			
	2 3 4 6			
	1690			



XP.NO:5(c)	LONGEST COMMON SUBSEQUENCE	
ATE:		
:		

Given two strings find the length of the common longest subsequence(need not be contiguous) between the two.

Example: s1:

ggtabe s2:

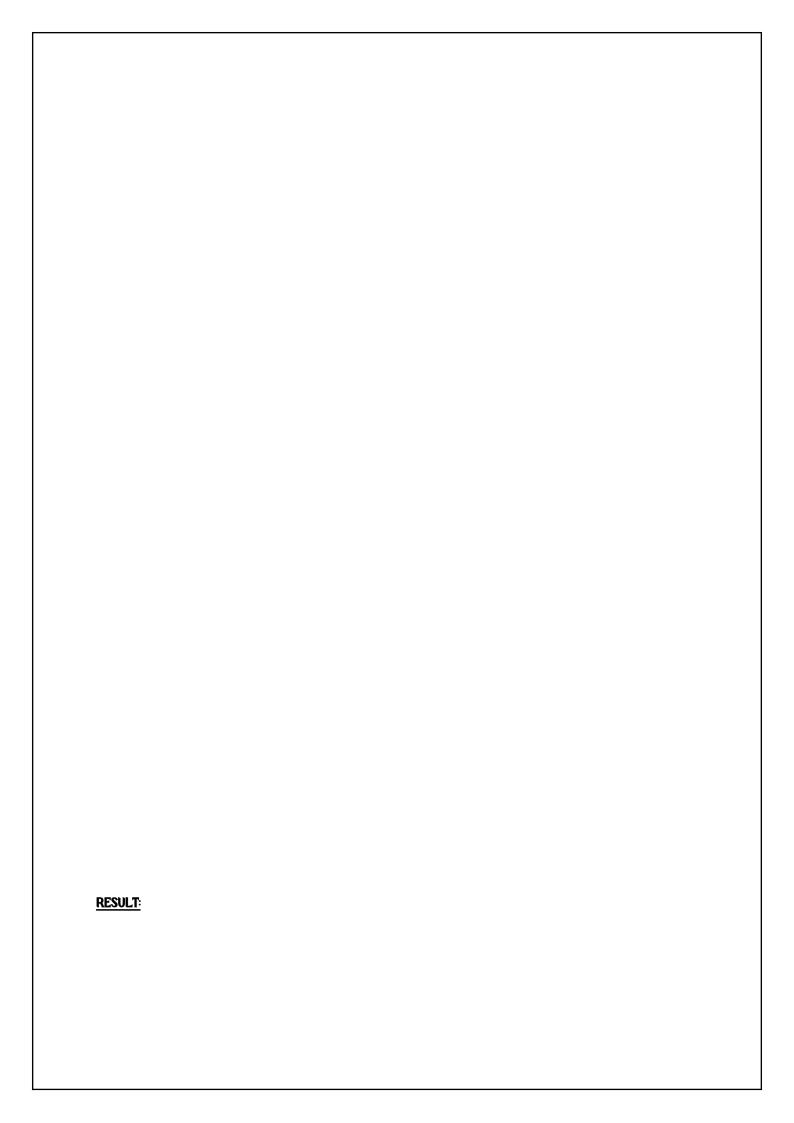
tgatasb

si a g g t a b $s2 \qquad \qquad g \qquad \qquad x \qquad \qquad t \qquad \qquad x \qquad \qquad a \qquad \qquad y \qquad \qquad b$

The length is 4

Solveing it using Dynamic Programming

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



EXP.NO:5(d)	LONGEST NON-DECREASING SUBSEQUENCE
DATE:	LONGLOT NON DEGILLIGING GODDLYCENCE

AIM:

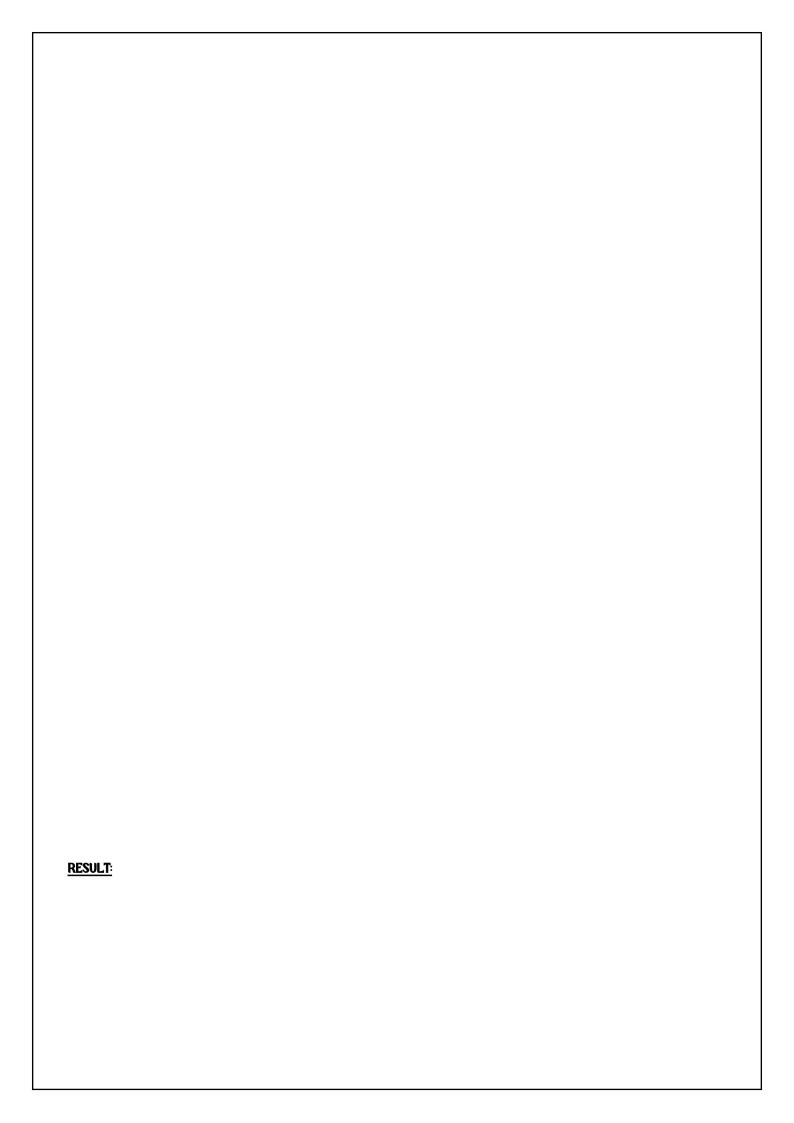
```
Problem statement:
Find the length of the Longest Non-decreasing Subsequence in a given Sequence. Eg:
Input:9
Sequence:[-1,3,4,5,2,2,2,2,3]
the subsequence is [-1,2,2,2,2,3]
Output:6
```

PROGRAM:

}

```
#include <stdio.h>
int longest_non_decreasing_subsequence(int arr[], int n) ( int dp[n];
   int max_length = 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 (arr[j] <= arr[i]) {
             dp[i] = (dp[i] > dp[j] + 1) ? dp[i] : dp[j] + 1;)) if (dp[i] >
      max_length) {
         max_length = dp[i]:
      }
   }
   return max_length;
int main() {
```

		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	~



EXP.NO:6(a)	FINDING DUPLICATES-O(N'2) TIME COMPLEXITY,O(1) SPACE COMPLEXITY
DATE:	

AIM:

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
11234	

```
}

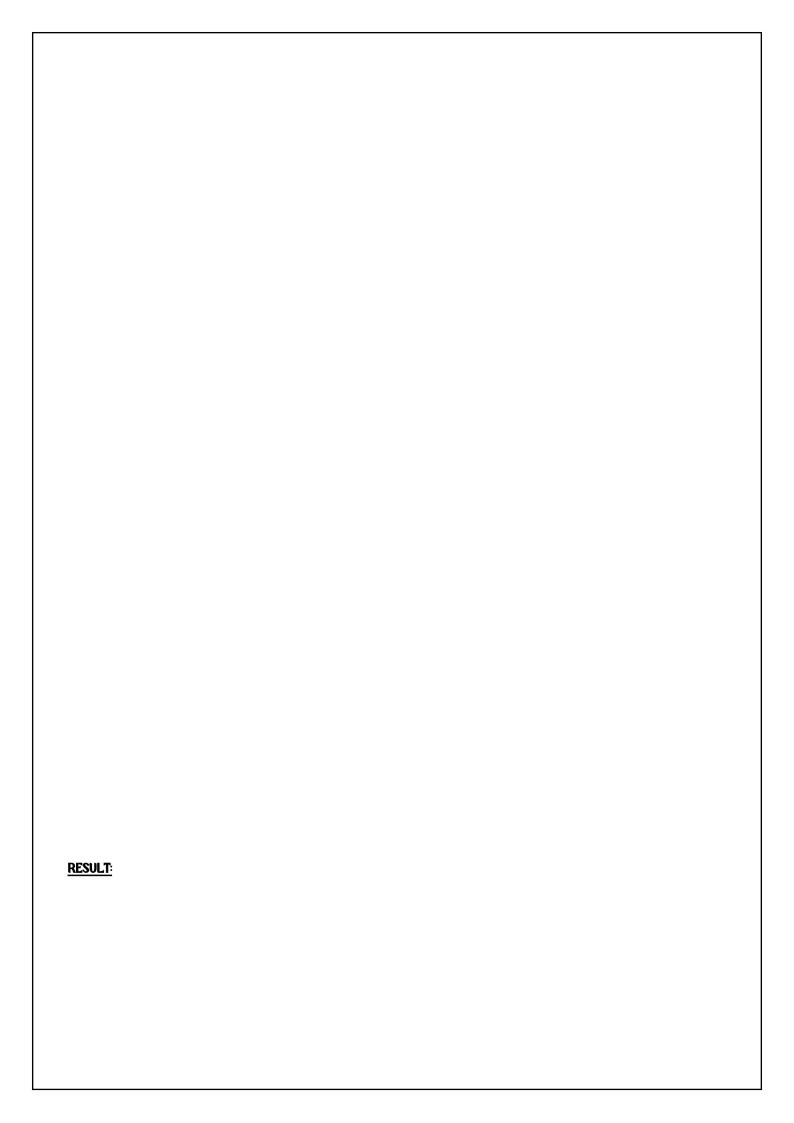
return slow:

}

int main() ( int n:
    scanf("%d", &n): int
    arr[n]:
    for (int i = 0: i < n: i++)
        ( scanf("%d", &arr[i]):
    }

int duplicate = findDuplicate(arr, n): printf("%d",
    duplicate):
    return 0:
}
</pre>
```

	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	~



EXP.NO:6(b)	FINDING DUPLICATES-O(N) TIME COMPLEXITY,O(1) SPACE COMPLEXITY
DATE:	

AIM:

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:

Result
1

```
}

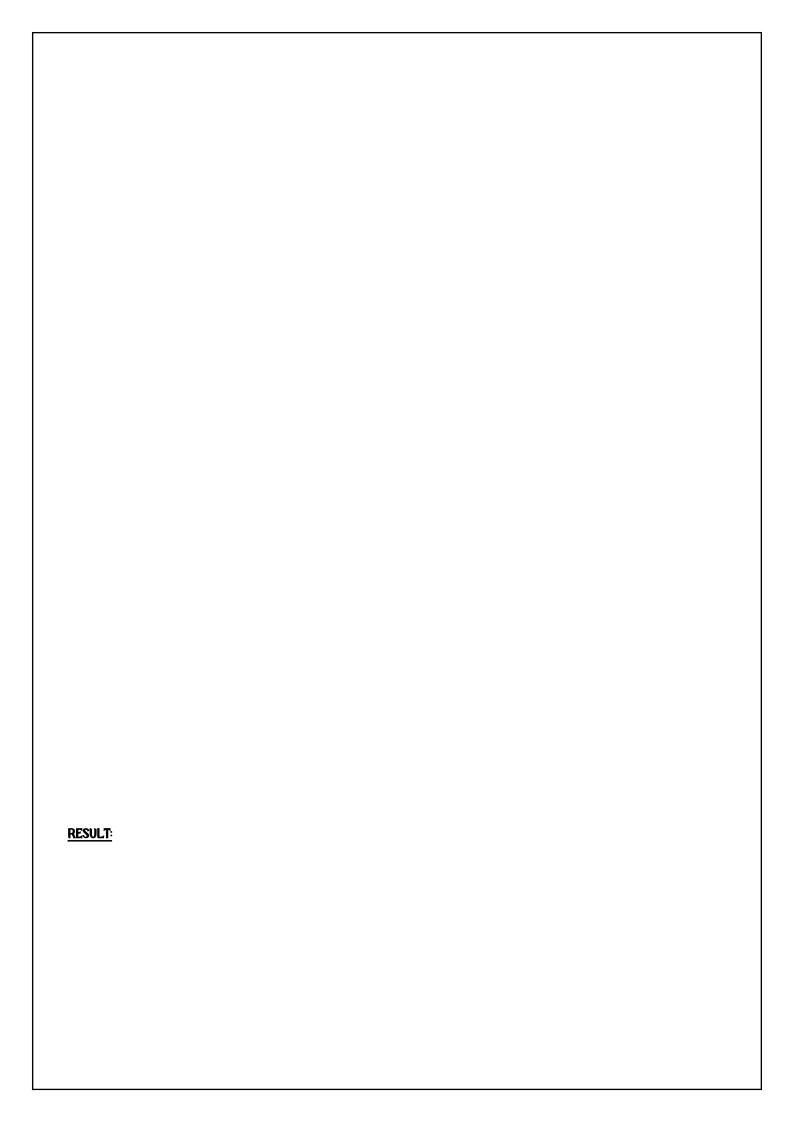
return slow:

}

int main() ( int n:
    scanf("%d", &n): int
    arr[n]:
    for (int i = 0: i < n: i++)
        ( scanf("%d", &arr[i]):
    }

int duplicate = findDuplicate(arr, n): printf("%d",
    duplicate):
    return 0:
}
</pre>
```

	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	~



EXP.NO:6(c)			
DATE:			
<u>IM</u> :			
<u>LGORITHM:</u>			

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

6123456

216

Output:

16

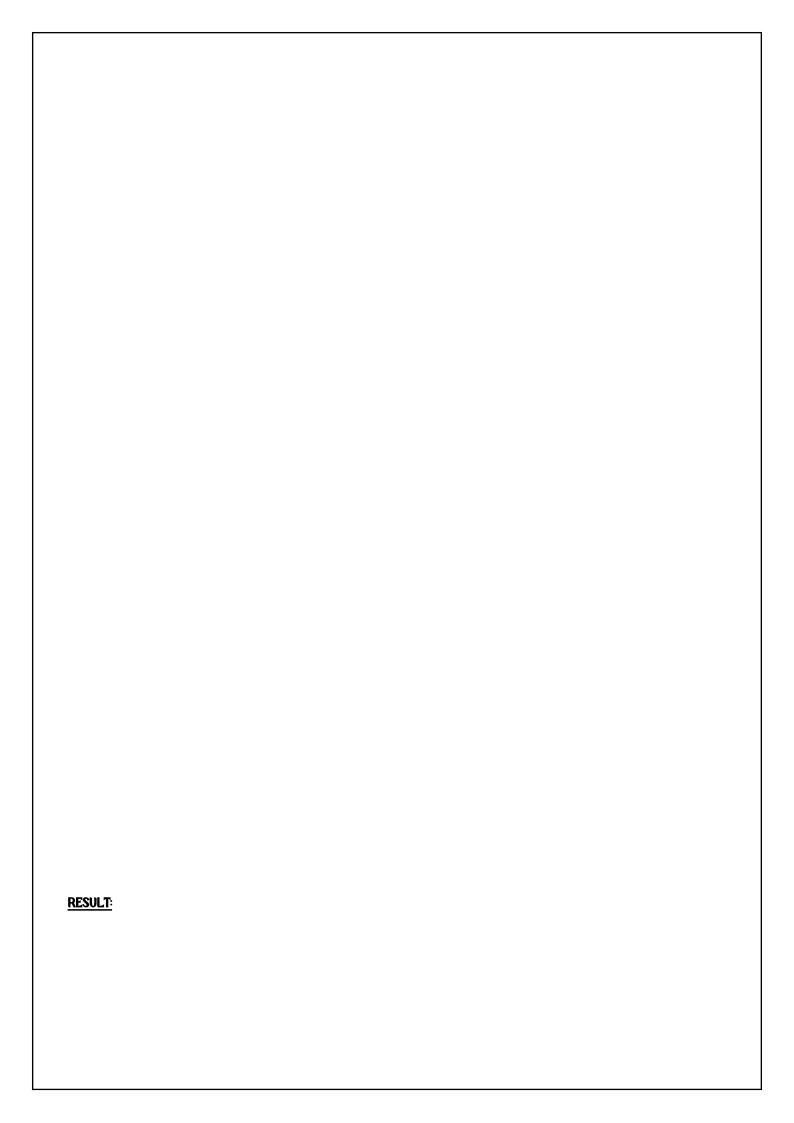
For example:

Input	Result
1	10 57
3 10 17 57	

Result

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

	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	*



EXP.NO:6(d)	PRINT INTERSECTION OF 2 SORTED ARRAYS-O(M+N)TIME COMPLEXITY,O(1) SPACE
DATE:	COMPLEXITY

<u>AIM</u>:

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

6123456

216

Output:

16

For example:

Input	Result
1	10 57

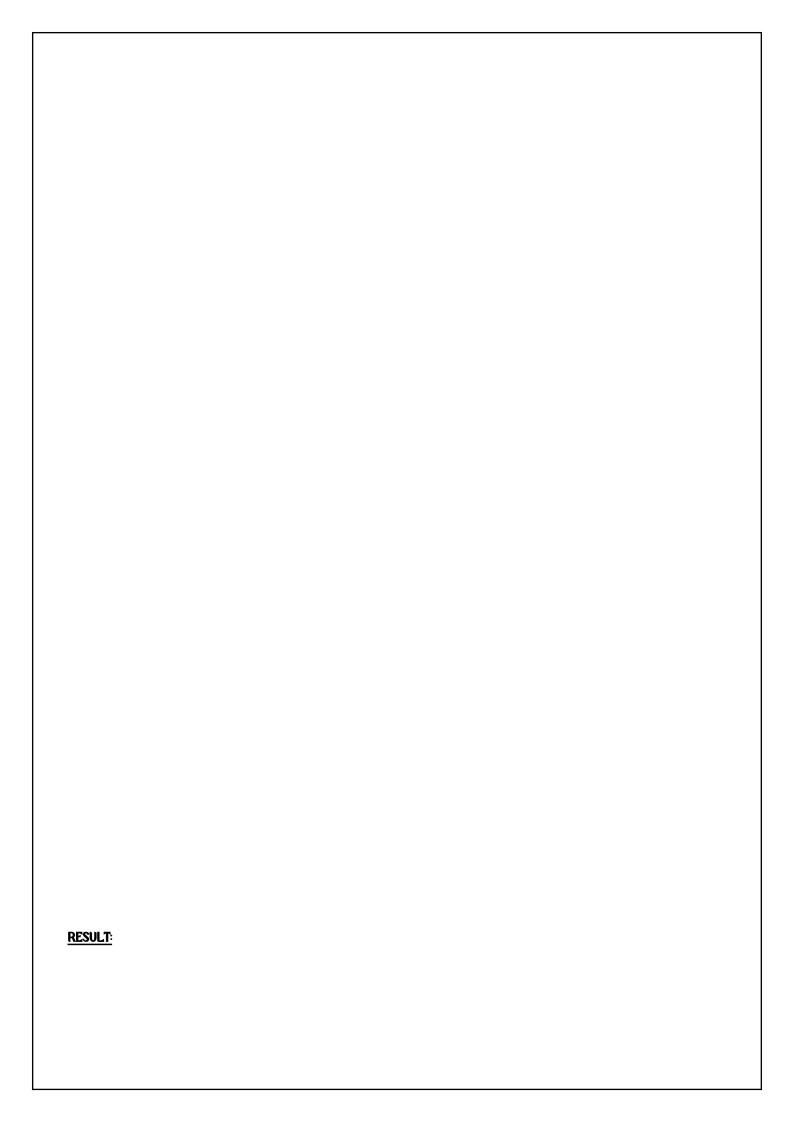
Input	Result
3 10 17 57	
6	
2 7 10 15 57 246	

```
#include <stdio.h>
void findIntersection(int arr1[], int n1, int arr2[], int n2) { int i = 0, j = 0:
   int found = 0:
   while (i < n1 && j < n2) { if (arr1[i]
      < arr2[j]) {</pre>
          j++;
      ) else if (arr2[j] < arr1[i]) { j++;
      } else {
         printf("%d ", arr1[i]); found = 1;
          j++;
         j++;
      }
   }
   if (Ifound) {
      printf("No Intersection");
   }
```

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



EXP.NO:6(e)	PAIR WITH DIFFERENCE-O(N'2)TIME COMPLEXITY,O(1) SPACE COMPLEXITY
DATE:	

<u>AIM</u>:

ALGORITHM:

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 = 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:

Result
1

PROGRAM:

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

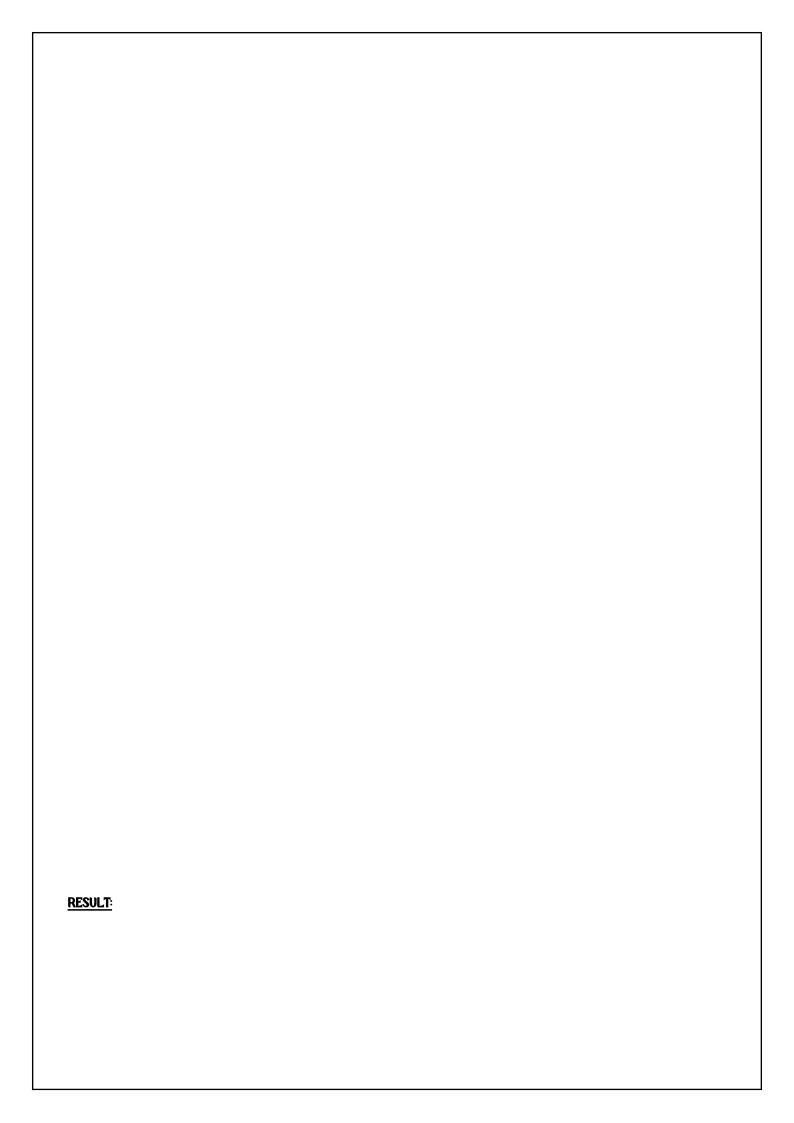
int findPairWithDifference(int arr[], int n, int k) { int i = 0, j =

1: while (i < n && j < n) {

```
int diff = arr[j] - arr[i]; if (i !=
      j \&\& diff == k) {
          return 1;
      }
      else if (diff < k) { j++;
      }
        else {
            j++;
      }
   }
   return 0;
}
int main() { int n,
   k;
   scanf("%d", &n); int
   arr[n];
   for (int i = 0; i < n; i++)
      { scanf("%d", & arr[i]);
   }
   scanf("%d", &k);
   int result = findPairWithDifference(arr, n, k); printf("%d\n",
   result);
   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	*



EXP.NO:6(f)	LONGEST NON-DECREASING SURSEQUENCE
DATE:	LONGEST NON-DECREASING SUBSEQUENCE

<u>AIM</u>:

ALGORITHM:

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[i] - 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
135	
4	

PROGRAM:

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

int findPairWithDifference(int arr[], int n, int k) { int i = 0, j =

1;

while (j < n) {

```
int diff = arr[j] - arr[i]; if (i !=
      j \&\& diff == k) {
          return 1:
       }
       else if (diff < k) { j++;
       }
       else {
          j++;
          if (i == j) { j++:
          }
       }
   }
   return 0;
}
int main() { int n,
   k;
   scanf("%d", &n); int
   arr[n];
   for (int i = 0; i < n; i++)
      { scanf("%d", &arr[i]);
   }
   scanf("%d", &k);
   int result = findPairWithDifference(arr, n, k); printf("%d\n",
   result);
   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	~

