RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

RAJALAKSHMI
ENGINEERING COLLEGE

CS23331 - DESIGN AND ANALYSIS OF ALGORITHM
LABORATORY LAB MANUAL

Name : **LOKESH C** . . . . . .

Year / Branch / Section : **II YEAR / AIML / A** . . . . . .

Register No. : **231501086** . . . . . .

Semester : **III SEMESTER** . . . . . .

Academic Year : **2024-2025** . . . . . .

INDEX

REG. NO : 231501086 **NAME: LOKESH C**

YEAR : II YEAR **BRANCH: AIML** **SEC : B**

S. NO.	DATE	TITLE	PAGE NO.	TEACHER'S SIGNATURE / REMARKS
WEEK 01 – BASIC C PROGRAMS				
1.1		SWAPPING OF TWO NUMBERS		
1.2		ELIGIBILITY CRITERIA		
1.3		GROCERY ITEMS		
1.4		BABA'S GIVING PATTERN		
1.5		PUNCTUALITY INCENTIVE		
1.6		DIVISIBILITY FINDER		
1.7		QUOTIENT AND REMAINDER		
1.8		GREATEST OF ALL NUMBERS		
1.9		EVEN OR ODD		
1.10		FACTORIAL OF A NUMBER		
1.11		SUM OF N NATURAL NUMBERS		
1.11		FIBONACCI SERIES		
1.12		POWER OF INTEGERS		
1.13		PRIME OR NON PRIME		
1.14		REVERSE OF AN INTEGER		
WEEK 02 - FINDING TIME COMPLEXITY OF ALGORITHMS				
2.1		COUNTER METHOD-WHILE LOOP		
2.2		COUNTER METHOD-FOR LOOP		
2.3		COUNTER METHOD-FACTORS		
2.4		COUNTER METHOD-FUNCTION		
2.5		COUNTER METHOD-REVERSE		

WEEK 03 – DIVIDE AND CONQUER				
3.1		**NUMBER OF ZEROS IN AN ARRAY**		
3.2		**MAJORITY ELEMENT**		
3.3		**FINDING FLOOR VALUE**		
3.4		**TWO ELEMENTS SUM TO X**		
3.5		**IMPLEMENTATION OF QUICK SORT**		
WEEK 04 – GREEDY ALGORITHMS				
4.1		**COIN PROBLEM**		
4.2		**COOKIES PROBLEM**		
4.3		**BURGER PROBLEM**		
4.4		**ARRAY SUM MAX PROBLEM**		
4.5		**PRODCUT OF ARRAY ELEMENTS-MIN**		
WEEK 05 – DYNAMIC PROGRAMMING				
5.1		**PLAYING WITH NUMBERS**		
5.2		**PLAYING WITH CHESSBOARD**		
5.3		**LONGEST COMMON SUBSEQUENCE**		
5.4		**LONGEST NON-DECREASING SUBSEQUENCE**		
WEEK 06 – COMPETITIVE PROGRAMMING				
6.1		**FINDING DUPLICATES-O(N^2) TIME COMPLEXITY,O(1) SPACE COMPLEXITY**		
6.2		**FINDING DUPLICATES-O(N) TIME COMPLEXITY,O(1) SPACE COMPLEXITY**		
6.3		**PRINT INTERSECTION OF 2 SORTED ARRAYS-O(M*N)TIME COMPLEXITY,O(1) SPACE COMPLEXITY**		
6.4		**PRINT INTERSECTION OF 2 SORTED ARRAYS-O(M+N)TIME COMPLEXITY,O(1) SPACE COMPLEXITY**		
6.5		**PAIR WITH DIFFERENCE-O(N^2)TIME COMPLEXITY,O(1) SPACE COMPLEXITY**		
6.6		**PAIR WITH DIFFERENCE -O(N) TIME COMPLEXITY,O(1) SPACE COMPLEXITY**		

WEEK 01 – BASIC C PROGRAMS

EXPERIMENT NO : 1.1 **DATE :**

REGISTER NO : 231501086 **NAME : LOKESH C**

SWAPPING OF TWO NUMBERS

GIVEN TWO NUMBERS , WRITE A C PROGRAM TO SWAP THE NUMBERS.

FOR EXAMPLE

Input	Result
10 20	20 10

PROGRAM

```c
#include<stdio.h>
int main()
{
int a;
int b;
int temp;
scanf("%d %d",&a,&b);
/*swapping the two numbers*/
temp=a;
a=b;
b=temp;
printf("%d %d",a,b);
}
```

OUTPUT

	Input	Expected	Got	
✔	10 20	20 10	20 10	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

EXPERIMENT NO : 1.2 **DATE :**

REGISTER NO : 231501086 **NAME : LOKESH C**

ELIGIBILITY CRITERIA

WRITE A 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

TEST CASE 2 :

INPUT

50 80 80

OUTPUT

THE CANDIDATE IS ELIGIBLE

TEST CASE 3

INPUT

50 60 40

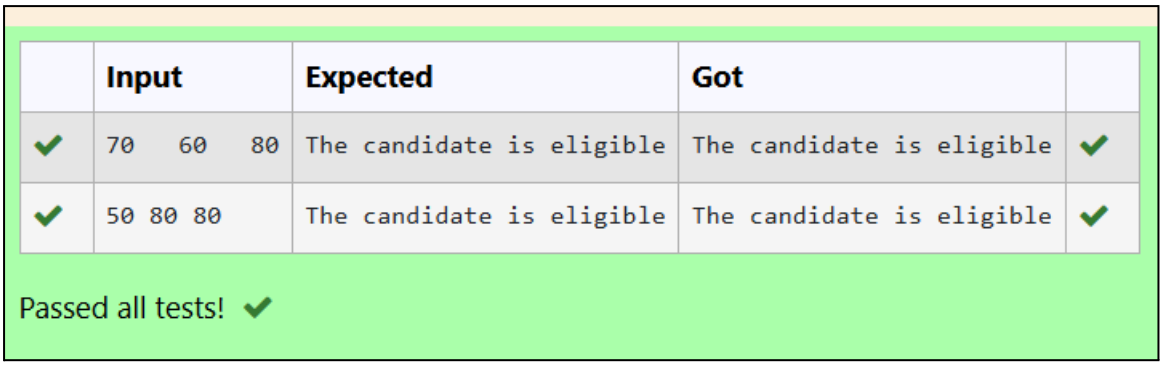
OUTPUT

THE CANDIDATE IS NOT ELIGIBLE

PROGRAM

```c
#include<stdio.h>

int main()

{
    int mark1;
    int mark2;
    int mark3;
    int total;
    scanf ("%d %d %d",&mark1,&mark2,&mark3);
    total=mark1+mark2+mark3;

    if(mark1>=65 && mark2>=55 && mark3>=50 && total>=180)
    {
        printf("The candidate is eligible");
    }
    else if(total>=180)
    {
        printf("The candidate is eligible");
    }
    else{
        printf("The candidate is not eligible");
    }
}
```

OUTPUT



	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! ✔

EXPERIMENT NO : 1.3 **DATE :**

REGISTER NO : 231501086 **NAME : LOKESH C**

GROCERY ITEMS

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

PROGRAM

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

    int discount;
    scanf("%d",&b);
    if(b>2000)
    {
        discount=b*0.10;

        printf("%d",b-discount);
    }
    else
    printf("%d",b);
}
```

OUTPUT

	Input	Expected	Got	
✔	1900	1900	1900	✔
✔	3000	2700	2700	✔

Passed all tests! ✔

EXPERIMENT NO : 1.4 **DATE :**

REGISTER NO : 231501086 **NAME : LOKESH C**

BABA'S GIVING PATTERN

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

EXPLANATION:

Baba donated to two beggars. So when he encountered second beggar he had 100*2 = Rs.200 and when he encountered 1st he had 200*2 = Rs.400.

PROGRAM

```c
#include<stdio.h>
int main()
{
    int money;
    int beggar;
    int amount;
    scanf("%d %d",&money,&beggar);

    amount=money*beggar*2;
    printf("%d",amount);
}
```

OUTPUT

	Input	Expected	Got	
✔	100 2	400	400	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

EXPERIMENT NO : 1.5 **DATE :**

REGISTER NO : 231501086 **NAME : LOKESH C**

PUNCTUALITY INCENTIVE

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

EXPLANATION:

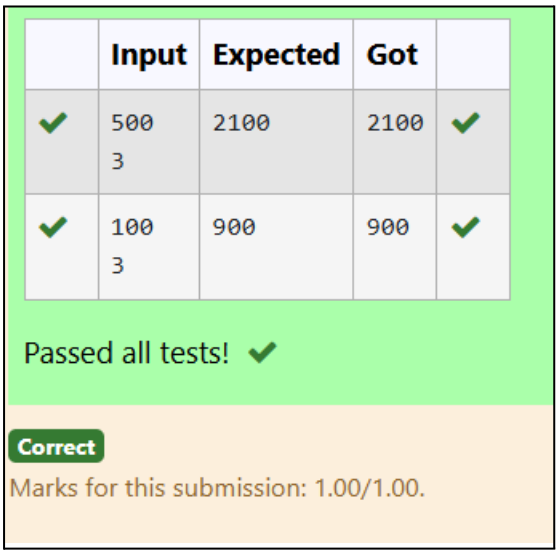
ON MONDAY THE EMPLOYEE RECEIVES RS.500, ON TUESDAY RS.700, ON WEDNESDAY RS.900

SO TOTAL = RS.2100

PROGRAM

```c
#include<stdio.h>
int main()
{
  int a,b,sum=0;
  scanf("%d",&a);
  scanf("%d",&b);
  for(int i=0;i<b;i++)
  {
      sum+=a;
      a=a+200;

  }
  printf("%d",sum);
}
```

OUTPUT



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

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

EXPERIMENT NO : 1.6 **DATE :**

REGISTER NO : 231501086 **NAME : LOKESH C**

DIVISIBILITY FINDER

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

EXAMPLE INPUT/OUTPUT 1:

INPUT:
2
40
7

OUTPUT:
35 28 21 14 7

EXAMPLE INPUT/OUTPUT 2:

INPUT:
66
121
11

OUTPUT:

121 110 99 88 77 66

PROGRAM

```c
#include<stdio.h>
int main()
{
    int m;
    int n;
    int x;
    scanf("%d %d",&m,&n);
    scanf("%d",&x);
    for(int i=n;i>m-1;i--)
    {
        if(i%x==0){
            printf("%d ",i);
        }
    }
}
```

OUTPUT

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

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

EXPERIMENT NO : 1.7 **DATE :**

REGISTER NO : 231501086 **NAME : LOKESH C**

QUOTIENT & REMAINDER

WRITE A C PROGRAM TO FIND THE QUOTIENT & REMAINDER OF GIVEN INTEGERS

FOR EXAMPLE

Input	Result
12 3	4 0

PROGRAM

```c
#include<stdio.h>
int main()
{
  int dd;
  int dr;
  scanf("%d",&dd);
  scanf("%d",&dr);
  int q;
  int rem;
  q=dd/dr;
  printf("%d\n",q);
  rem=dd%dr;
  printf("%d\n",rem);
}
```

OUTPUT

	Input	Expected	Got	
✔	12 3	4 0	4 0	✔

Passed all tests! ✔

EXPERIMENT NO : 1.8 **DATE :**

REGISTER NO : 231501086 **NAME : LOKESH C**

GREATEST OF ALL NUMBERS

WRITE A C PROGRAM TO FIND THE GREATEST NUMBERS OF 3 INTEGERS.

FOR EXAMPLE

Input	Result
10 20 30	30

PROGRAM

```c
#include<stdio.h>
int main()
{
    int a;
    int b;
    int c;
    scanf("%d %d %d",&a,&b,&c);

    if(a>b && a>c){
        printf("%d",a);
    }
    else if(b>c && b>a){
        printf("%d",b);
    }
    else
    printf("%d",c);
}
```

OUTPUT

	Input	Expected	Got	
✔	10 20 30	30	30	✔

Passed all tests! ✔

EXPERIMENT NO : 1.9 **DATE :**

REGISTER NO : 231501086 **NAME : LOKESH C**

EVEN OR ODD

WRITE A C PROGRAM TO FIND THE NUMBER IS ODD OR EVEN ?

FOR EXAMPLE

Input	Result
12	Even
11	Odd

PROGRAM

```c
#include<stdio.h>
int main()
{
    int a;
    scanf("%d",&a);

    if(a%2==0){
        printf("Even");
    }
    else
    printf("Odd");
}
```

OUTPUT

	Input	Expected	Got	
✔	12	Even	Even	✔
✔	11	Odd	Odd	✔

Passed all tests! ✔

EXPERIMENT NO : 1.10 **DATE :**

REGISTER NO : 231501086 **NAME : LOKESH C**

FACTORIAL OF A NUMBER

WRITE A PROGRAM TO FIND THE FACTORIAL OF A NUMBER

FOR EXAMPLE

Input	Result
5	120

PROGRAM

```c
#include<stdio.h>
int main()
{
    int factorial;
    factorial=1;
    int n;
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
    {
        factorial=factorial*i;
    }
    printf("%d",factorial);
}
```

OUTPUT

	Input	Expected	Got	
✔	5	120	120	✔

Passed all tests! ✔

EXPERIMENT NO : 1.11 **DATE :**

REGISTER NO : 231501086 **NAME : LOKESH C**

SUM OF N NATURAL NUMBERS

WRITE A C PROGRAM TO FIND THE SUM OF N NATURAL NUMBERS

FOR EXAMPLE

Input	Result
3	6

PROGRAM

```c
#include<stdio.h>
int main(){
    int number;
    scanf("%d",&number);
    int i;
    int sum;
    sum=0;
    for(i=number;i>=0;i--)
    {
        sum=sum+i;
    }
    printf("%d",sum);
}
```

OUTPUT

	Input	Expected	Got	
✔	3	6	6	✔

Passed all tests! ✔

EXPERIMENT NO : 1.12 **DATE :**

REGISTER NO : 231501086 **NAME : LOKESH C**

FIBONACCI SERIES

WRITE A C PROGRAM TO FIND THE NTH TERM OF FIBONACCI SERIES

FOR EXAMPLE

Input	Result
0	0
1	1
4	3

PROGRAM

```c
#include<stdio.h>
int main()
{
 int a;
 int b;
 int c;
 int sum;
 b=0;
 c=1;
 sum=0;
 scanf("%d",&a);
 for(int i=0;i<a-1;i++){
     sum=b+c;
     b=c;
     c=sum;
 }
 if(a==1){
     printf("1");
 }else{
     printf("%d",sum);
 }
 }
```

OUTPUT

	Input	Expected	Got	
✔	0	0	0	✔
✔	1	1	1	✔
✔	4	3	3	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

EXPERIMENT NO : 1.13 **DATE :**

REGISTER NO : 231501086 **NAME : LOKESH C**

POWER OF INTEGERS

WRITE A C PROGRAM TO FIND THE POWER OF INTEGERS.

INPUT:

A B

OUTPUT:

A^B VALUE

FOR EXAMPLE

Input	Result
2 5	32

PROGRAM

```c
#include<stdio.h>
#include<math.h>
int main()
{
    int a;
    int b;
    scanf("%d %d",&a,&b);

    int power;
    power=pow(a,b);
    printf("%d",power);
}
```

OUTPUT

	Input	Expected	Got	
✔	2 5	32	32	✔

Passed all tests! ✔

EXPERIMENT NO : 1.14 **DATE :**

REGISTER NO : 231501086 **NAME : LOKESH C**

PRIME OR NON PRIME

WRITE A C PROGRAM TO FIND WHETHER NUMBER IS PRIME OR NOT?

FOR EXAMPLE

Input	Result
7	Prime
9	No Prime

PROGRAM

```c
#include<stdio.h>
int main()
{
    int number;
    scanf("%d",&number);

    if(number%2==0){
    printf("No Prime");
    }
    else if(number%3==0){
        printf("No Prime");
    }
    else if(number%number==0 && number/number==1){
        printf("Prime");
    }
    else
    printf("Prime");

}
```

OUTPUT

	Input	Expected	Got	
✔	7	Prime	Prime	✔
✔	9	No Prime	No Prime	✔

Passed all tests! ✔

EXPERIMENT NO : 1.15 **DATE :**

REGISTER NO : 231501086 **NAME : LOKESH C**

REVERSE OF AN INTEGER

WRITE A C PROGRAM TO FIND THE REVERSE OF AN INTEGER .

PROGRAM

```c
#include<stdio.h>
int main()
{
    int n;
    scanf("%d",&n);
    int reverse;
    reverse=0;
    int last;
    last=0;
    while(n!=0){
    last=n%10;
    reverse=reverse*10+last;
    n/=10;
    }
    printf("%d",reverse);
}
```

OUTPUT

	Input	Expected	Got	
✔	123	321	321	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

WEEK 02 - FINDING TIME

COMPLEXITY OF ALGORITHMS

EXPERIMENT NO : 2.1 **DATE :**

REGISTER NO : 231501086 **NAME : LOKESH C**

COUNTER METHOD-WHILE LOOP

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

PROGRAM

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

OUTPUT

EXPERIMENT NO : 2.2 **DATE :**

REGISTER NO : 231501086 **NAME : LOKESH C**

COUNTER METHOD-FOR LOOP

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

```c
#include<stdio.h>
int main()
{
        int count=0;
        int n;
        scanf("%d",&n);
        if(n==1){
            count++;
            //printf("*");
        }
        //count++;
        else{
            count++;
            for(int i=1;i<=n;i++)
            {
                count++;
                for(int j=1;j<=n;j++)
                {
                    count++;
                    //printf("*");
                    count++;
                    //printf("*");
                    count++;
                    break;
                    count++;
                }
                count++;
            }count++;
        }
        printf("%d",count);
    }
```

OUTPUT

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

Passed all tests! ✔

EXPERIMENT NO : 2.3 **DATE :**

REGISTER NO : 231501086 **NAME : LOKESH C**

COUNTER METHOD-FACTORS

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

PROGRAM

```c
#include<stdio.h>
int main()
{
    int num;
    scanf("%d",&num);
    int count=0;
    int i;
    for(i=1;i<=num;i++)
    {
        count++;
        if(num%i==0)
        {
            count++;
            //printf("%d ",i);
            //count++;
        }count++;
    }count++;
    printf("%d",count);
}
```

OUTPUT

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

Passed all tests! ✔

EXPERIMENT NO : 2.4 **DATE :**

REGISTER NO : 231501086 **NAME : LOKESH C**

COUNTER METHOD-FUNCTION

CONVERT THE FOLLOWING ALGORITHM INTO A PROGRAM AND FIND ITS TIME COMPLEXITY USING COUNTER METHOD.

```
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++;

}
```

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

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

OUTPUT

	Input	Expected	Got	
✔	4	30	30	✔
✔	10	212	212	✔

Passed all tests! ✔

EXPERIMENT NO : 2.5 **DATE :**

REGISTER NO : 231501086 **NAME : LOKESH C**

COUNTER METHOD-REVERSE

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

PROGRAM

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

OUTPUT

	Input	Expected	Got	
✔	12	11	11	✔
✔	1234	19	19	✔

Passed all tests! ✔

WEEK 03 – DIVIDE AND CONQUER

EXPERIMENT NO : 3.1 **DATE :**

REGISTER NO : 231501086 **NAME : LOKESH C**

NUMBER OF ZEROS IN AN ARRAY

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.

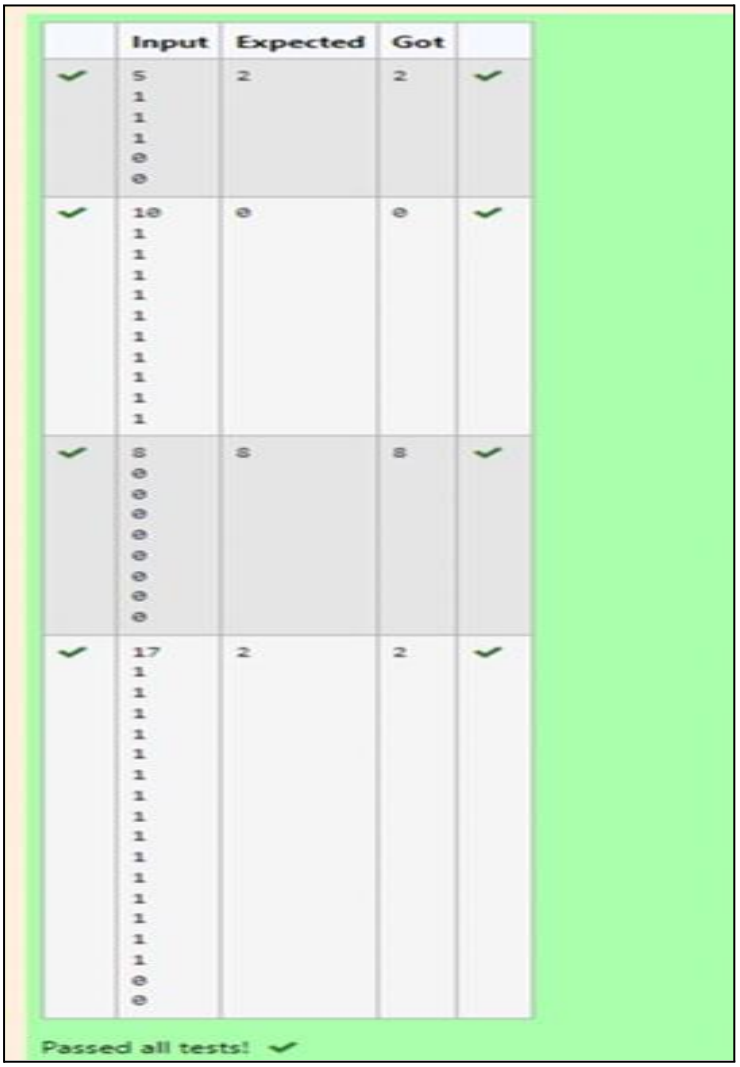
PROGRAM

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

```
    {
```

```
      if(arr[i]==0)
      {
            count=count+1;
    }
    }
    printf("%d",count);

}
```

OUTPUT



	Input	Expected	Got	
✔	5 1 1 1 0 0	2	2	✔
✔	10 1 1 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 1 1 1 0 0	2	2	✔

Passed all tests! ✔

EXPERIMENT NO : 3.2 **DATE :**

REGISTER NO : 231501086 **NAME : LOKESH C**

MAJORITY ELEMENT

GIVEN AN ARRAY NUMS OF SIZE N, RETURN THE MAJORITY ELEMENT.

THE MAJORITY ELEMENT IS THE ELEMENT THAT APPEARS MORE THAN ⌊N / 2⌋ 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

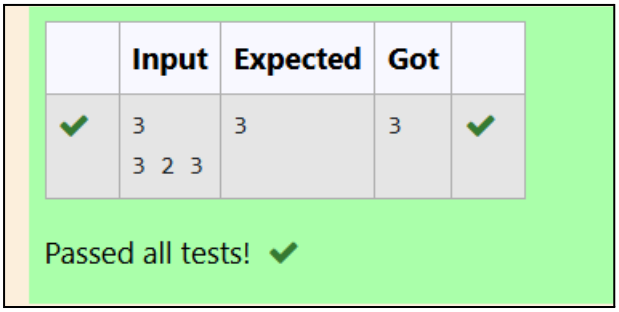
FOR EXAMPLE:

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

PROGRAM

```c
#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++){
        int count=0;
        for(int j=0;j<n;j++){
            if(a[i]==a[j]){
                count++;
            }
        }
        if(count>n/2){
            printf("%d",a[i]);
            break;
        }
    }
}
```

OUTPUT



	Input	Expected	Got	
✔	3 3 2 3	3	3	✔

Passed all tests! ✔

EXPERIMENT NO : 3.3 **DATE :**

REGISTER NO : 231501086 **NAME : LOKESH C**

FINDING FLOOR VALUE

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

PROGRAM

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

```
            floor=arr[j];
        }
        printf("%d",floor);
}
```

OUTPUT

	Input	Expected	Got	
✔	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! ✔

EXPERIMENT NO : 3.4 **DATE :**

REGISTER NO : 231501086 **NAME : LOKESH C**

TWO ELEMENTS SUM TO X

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”)

PROGRAM

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

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

    int i,j;
```

```
    int flag;
    int x;
    scanf("%d",&x);

    for(i=0;i<n;i++){

        for(j=i+1;j<n;j++){

            if(arr[i]+arr[j]==x){

                printf("%d\n%d",arr[i],arr[j]);

                flag=1;

                break;
            }

        }
    }
    if(flag==0)
    printf("No");
}
```

OUTPUT

	Input	Expected	Got	
✔	4 2 4 8 10 14	4 10	4 10	✔
✔	5 2 4 6 8 10 100	No	No	✔

Passed all tests! ✔

EXPERIMENT NO : 3.5 **DATE :**

REGISTER NO : 231501086 **NAME : LOKESH C**

IMPLEMENTATION OF QUICK SORT

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 67 34 12 98 78	12 34 67 78 98

PROGRAM

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

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

    for(int i = 0; i < n-1; i++) {
```

```
        for(int j = 0; j < n-i-1; j++)
{
            if(arr[j] > arr[j+1]) {
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }


    for(int i = 0; i < n; i++)
        printf("%d ", arr[i]);
    }

    return 0;
}
```

OUTPUT

	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! ✔

WEEK 04 – GREEDY ALGORITHMS

EXPERIMENT NO : 4.1 **DATE :**

REGISTER NO : 231501086 **NAME : LOKESH C**

COIN PROBLEM

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

EXPLANATON:

WE NEED A 50 RS NOTE AND A 10 RS NOTE AND TWO 2 RUPEE COINS.

PROGRAM

```c
#include<stdio.h>
int main()
{
    int value;
    scanf("%d",&value);

    int currency[]={1000,500,100,50,20,10,5,2,1};
    int totalcurrency;
    totalcurrency=sizeof(currency)/sizeof(currency[0]);

    int count=0;

    for(int i=0;i<totalcurrency;i++)
    {
        if(value==0)
        {
            break;
        }
        count=count+(value/currency[i]);

        value=value%currency[i];
    }
    printf("%d",count);
}
```

OUTPUT

	Input	Expected	Got	
✔	49	5	5	✔

Passed all tests! ✔

EXPERIMENT NO : 4.2 **DATE :**

REGISTER NO : 231501086 **NAME : LOKESH C**

COOKIES PROBLEM

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

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 <= G.LENGTH <= 3 * 10^4

0 <= S.LENGTH <= 3 * 10^4

1 <= G[I], S[J] <= 2^31 - 1

PROGRAM

```c
#include <stdio.h>
int main() {
    int n;
    scanf("%d", &n);
    int greedfactor[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &greedfactor[i]);
    }
    int  m;
    scanf("%d", &m);
    int cookiesize[m];
    for (int j = 0; j < m; j++) {
        scanf("%d", &cookiesize[j]);
    }
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (greedfactor[j] > greedfactor[j + 1]) {
                int temp = greedfactor[j];
                greedfactor[j] = greedfactor[j + 1];
                greedfactor[j + 1] = temp;
            }
        }
    }
    for (int i = 0; i < m - 1; i++) {
        for (int j = 0; j < m - i - 1; j++) {
            if (cookiesize[j] > cookiesize[j + 1]) {
                int temp = cookiesize[j];
                cookiesize[j] = cookiesize[j + 1];
                cookiesize[j + 1] = temp;
            }
        }
    }
    int i = 0;
    int j = 0;
    int contents = 0;
    while (i < n && j < m) {
        if (cookiesize[j] >= greedfactor[i]) {
            contents++;
            i++;
        }
        j++;
    }
    printf("%d\n", contents);
    return 0;
}
```

OUTPUT

	Input	Expected	Got	
✔	2 1 2 3 1 2 3	2	2	✔

Passed all tests! ✔

EXPERIMENT NO : 4.3 **DATE :**

REGISTER NO : 231501086 **NAME : LOKESH C**

BURGER PROBLEM

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 3I * 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 (30 * 1) + (31 * 3) + (32 * 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

PROGRAM

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

OUTPUT

	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! ✔

EXPERIMENT NO : 4.4 **DATE :**

REGISTER NO : 231501086 **NAME : LOKESH C**

ARRAY SUM MAX PROBLEM

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

PROGRAM

```c
#include<stdio.h>
int main(){
    int n;
    scanf("%d",&n);
    int arr[n];
    for(int i=0;i<n;i++)
    {
        scanf("%d ",&arr[i]);
    }
    for(int i=0;i<n-1;i++)
    {
```

```
        for(int j=0;j<n-i-1;j++){
            if(arr[j]>arr[j+1]){
            int temp=arr[j];
            arr[j]=arr[j+1];
            arr[j+1]=temp;
        }
    }
}

int maximum=0;
for(int i=0;i<n;i++){
    maximum=maximum+(arr[i]*i);
}printf("%d\n",maximum);
}
```

OUTPUT

	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! ✔

EXPERIMENT NO : 4.5 **DATE :**

REGISTER NO : 231501086 **NAME : LOKESH C**

PRODCUT OF ARRAY ELEMENTS-MIN

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] * B[I]) FOR ALL I IS MINIMUM.

FOR EXAMPLE

Input	Result
3 1 2 3 4 5 6	28

PROGRAM

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

```
                int temp=arrayTwo[j];
                arrayTwo[j]=arrayTwo[j+1];
                arrayTwo[j+1]=temp;
            }
        }
    }
    int minimumsum = 0;
    for (int i = 0; i < n; i++) {
        minimumsum=minimumsum+arrayOne[i]*arrayTwo[i];
    }
    printf("%d\n", minimumsum);
}
```

OUTPUT

	Input	Expected	Got	
✔	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	✔

WEEK – 05

PLAYING WITH NUMBERS

EXPERIMENT NO : 5.1 **DATE :**

REGISTER NO : 231501086 **NAME : LOKESH C**

PLAYING WITH NUMBERS

PLAYING WITH NUMBERS:

RAM AND SITA ARE PLAYING WITH NUMBERS BY GIVING PUZZLES TO EACH OTHER. NOW IT WAS RAM TERM, SO HE GAVE SITA A POSITIVE INTEGER 'N' AND TWO NUMBERS 1 AND 3. HE ASKED HER TO FIND THE POSSIBLE WAYS BY WHICH THE NUMBER N CAN BE REPRESENTED USING 1 AND 3.WRITE ANY EFFICIENT ALGORITHM TO FIND THE POSSIBLE WAYS.

EXAMPLE 1:

INPUT:

6

OUTPUT:

6

EXPLANATION:

THERE ARE 6 WAYS TO 6 REPRESENT NUMBER WITH 1 AND 3

1+1+1+1+1+1
3+3
1+1+1+3
1+1+3+1
1+3+1+1
3+1+1+1

INPUT FORMAT

FIRST LINE CONTAINS THE NUMBER N

OUTPUT FORMAT

PRINT:

THE NUMBER OF POSSIBLE WAYS 'N' CAN BE REPRESENTED USING 1 AND 3

SAMPLE INPUT

6

SAMPLE OUTPUT

6

PROGRAM

```c
#include <stdio.h>
int main() {
    long n;
    scanf("%ld", &n);
    if (n < 0) {
        return 0;
    }
    long array[n + 1];
    array[0] = 1;
    array[1] = 1;
    array[2] = 1;
    array[3] = 2;
    for (long i = 4; i <= n; i++) {
        array[i] = array[i - 1] + array[i - 3];
    }
    printf("%ld\n", array[n]);
    return 0;
}
```

OUTPUT

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

Passed all tests! ✔

EXPERIMENT NO : 5.2 **DATE :**

REGISTER NO : 231501086 **NAME : LOKESH C**

PLAYING WITH CHESSBOARD

PLAYING WITH CHESSBOARD:

RAM IS GIVEN WITH AN N*N CHESSBOARD WITH EACH CELL WITH A MONETARY VALUE. RAM STANDS AT THE (0,0), THAT THE POSITION OF THE TOP LEFT WHITE ROOK. HE IS BEEN GIVEN A TASK TO REACH THE BOTTOM RIGHT BLACK ROOK POSITION (N-1, N-1) CONSTRAINED THAT HE NEEDS TO REACH THE POSITION BY TRAVELING THE MAXIMUM MONETARY PATH UNDER THE CONDITION THAT HE CAN ONLY TRAVEL ONE STEP RIGHT OR ONE STEP DOWN THE BOARD. HELP RAM TO ACHIEVE IT BY PROVIDING AN EFFICIENT DP ALGORITHM.

EXAMPLE:

INPUT

3

1 2 4

2 3 4

8 7 1

OUTPUT:

19

EXPLANATION:

TOTALLY THERE WILL BE 6 PATHS AMONG THAT THE OPTIMAL IS

OPTIMAL PATH VALUE:1+2+8+7+1=19

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

PROGRAM

```
#include <stdio.h>
int maxMonetaryPath(int n, int board[n][n])
{
    int dp[n][n];
    dp[0][0] = board[0][0];

    for (int j = 1; j < n; j++){
        dp[0][j] = dp[0][j - 1] + board[0][j];
    }

    for (int i = 1; i < n; i++) {
        dp[i][0] = dp[i - 1][0] + board[i][0];
    }

    for (int i = 1; i < n; i++) {

        for (int j = 1; j < n; j++) {

            dp[i][j] = board[i][j] + (dp[i - 1][j] > dp[i][j - 1] ? dp[i -
1][j] : dp[i][j - 1]);

        }
    }
    return dp[n - 1][n - 1];
}

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

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < n; j++) {

            scanf("%d", &board[i][j]);
        }
    }

    int maxValue = maxMonetaryPath(n, board);
    printf("%d\n", maxValue);
    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! ✔

EXPERIMENT NO : 5.3 **DATE :**

REGISTER NO : 231501086 **NAME : LOKESH C**

LONGEST COMMON SUBSEQUENCE

GIVEN TWO STRINGS FIND THE LENGTH OF THE COMMON LONGEST SUBSEQUENCE(NEED NOT BE CONTIGUOUS) BETWEEN THE TWO.

EXAMPLE:

S1: GGTABE

S2: TGATASB

S1: A G G T A B

S2: G X T X A Y B

THE LENGTH IS 4

SOLVING IT USING DYNAMIC PROGRAMMING

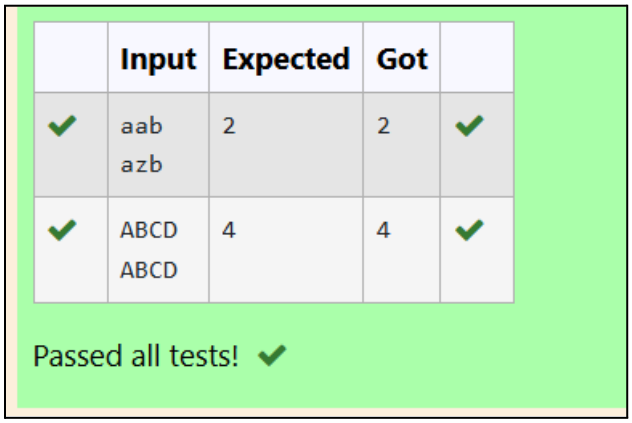
FOR EXAMPLE:

Input	Result
aab azb	2

PROGRAM

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

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

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

    for (int i = 0; i <= m; i++) {
        for (int j = 0; j <= n; j++) {
            if (i == 0 || j == 0) {
                dp[i][j] = 0;
            } else if (s1[i - 1] == s2[j - 1]) {
                dp[i][j] = dp[i - 1][j - 1] + 1;
            } else {
                dp[i][j] = (dp[i - 1][j] > dp[i][j - 1]) ? dp[i - 1][j] :
dp[i][j - 1];
            }
        }
    }

    return dp[m][n];
}

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

    fgets(s1, sizeof(s1), stdin);
    s1[strcspn(s1, "\n")] = '\0';

    fgets(s2, sizeof(s2), stdin);
    s2[strcspn(s2, "\n")] = '\0';
    int length = longestCommonSubsequence(s1, s2);
    printf("%d\n", length);

    return 0;
}
```

OUTPUT



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

Passed all tests! ✔

EXPERIMENT NO : 5.4 **DATE :**

REGISTER NO : 231501086 **NAME : LOKESH C**

LONGEST NON-DECREASING SUBSEQUENCE

PROBLEM STATEMENT:

FIND THE LENGTH OF THE LONGEST NON-DECREASING SUBSEQUENCE IN A GIVEN SEQUENCE.

EXAMPLE:

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 longseq(int arr[],int n){
        int dp[n];
        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[i] >= arr[j]) {
                dp[i] = (dp[i] > dp[j] + 1) ? dp[i] : dp[j] + 1;
            }
        }
```

```
}
    int maximumlength=0;
    for(int i=0;i<n;i++){
        if(dp[i]>maximumlength){
            maximumlength=dp[i];
        }
    }
    return maximumlength;
    }
    int main()
{
    int n;
    scanf("%d",&n);

    int arr[n];
    for(int i=0;i<n;i++)
    {
        scanf("%d",&arr[i]);
    }
    int length=longseq(arr, n);
    printf("%d\n",length);

    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! ✔

WEEK 06 – COMPETITIVE PROGRAMMING

EXPERIMENT NO : 6.1 **DATE :**

REGISTER NO : 231501086 **NAME :**

LOKESH C

FINDING DUPLICATES-O(N^2) TIME COMPLEXITY,O(1) SPACE COMPLEXITY

FIND DUPLICATE IN ARRAY.

- GIVEN A READ ONLY ARRAY OF N INTEGERS BETWEEN 1 AND N, FIND ONE NUMBER THAT REPEATS.

INPUT FORMAT:

- FIRST LINE - NUMBER OF ELEMENTS
- N LINES - N ELEMENTS

OUTPUT FORMAT:

ELEMENT X - THAT IS REPEATED

FOR EXAMPLE:

Input	Result
5 1 1 2 3 4	1

PROGRAM

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

```
    for(i=0;i<n;i++)
    {
        scanf("%d",&arr[i]);
    }
    for(i=0;i<n;i++){
        count=0;
        for(int j=0;j<n;j++){
            if(arr[i]==arr[j]){
                count=count+1;
            }
        }
    if(count>1){
        printf("%d\n",arr[i]);
        break;
    }
}
}
```

OUTPUT

	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! ✔

EXPERIMENT NO : 6.2 **DATE :**

REGISTER NO : 231501086 **NAME : LOKESH C**

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 2 3 4	1

PROGRAM

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

```
    for(i=0;i<n;i++){
        count=0;
        for(int j=0;j<n;j++){
            if(arr[i]==arr[j]){
                count=count+1;
            }
        }
    if(count>1){
        printf("%d\n",arr[i]);
        break;
    }
}

}
```

OUTPUT

	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! ✔

EXPERIMENT NO : 6.3 **DATE :**

REGISTER NO : 231501086 **NAME : LOKESH C**

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

FOR EXAMPLE:

Input	Result
1 3 10 17 57 6 2 7 10 15 57 246	10 57

PROGRAM

```c
#include <stdio.h>
void findIntersection(int arr1[], int v1, int arr2[], int v2) {
    int i = 0, j = 0;
    while (i < v1 && j < v2) {
        if (arr1[i] == arr2[j]) {
            printf("%d ", arr1[i]);
            i++;
            j++;
        } else if (arr1[i] < arr2[j]) {
            i++;
        } else {
            j++;
        }
    }
    printf("\n");
}
int main() {
    int T;
    scanf("%d", &T);
    while (T--) {
        int v1;
        scanf("%d", &v1);
        int arr1[v1];
        for (int i = 0; i < v1; i++) {
            scanf("%d", &arr1[i]);
        }
        int v2;
        scanf("%d", &v2);
        int arr2[v2];
        for (int i = 0; i < v2; i++) {
            scanf("%d", &arr2[i]);
        }
        findIntersection(arr1, v1, arr2, v2);
    }
    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! ✔

EXPERIMENT NO : 6.4 **DATE :**

REGISTER NO : 231501086 **NAME : LOKESH C**

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

FOR EXAMPLE:

Input	Result
1 3 10 17 57 6 2 7 10 15 57 246	10 57

PROGRAM

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

EXPERIMENT NO : 6.5 **DATE :**

REGISTER NO : 231501086 **NAME : LOKESH C**

PAIR WITH DIFFERENCE-O(N^2)TIME COMPLEXITY,O(1) SPACE COMPLEXITY

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

PROGRAM

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

```
    int n;
    scanf("%d",&n);
    int array[n];
    for(int i=0;i<n;i++)
    {
        scanf("%d",&array[i]);
    }
    int d;
    scanf("%d",&d);
    int count=0;
    for(int i=0;i<n;i++){
         for(int j=0;j<n;j++){
             if(i!=j){
                 if(array[j]-array[i]==d){
                     count=count+1;
                 }
             }
        }
    }
    if(count==0){
        printf("0");
    }else
     printf("1");
}
```

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! ✔

EXPERIMENT NO : 6.6 **DATE :**

REGISTER NO : 231501086 **NAME : LOKESH C**

PAIR WITH DIFFERENCE -O(N) TIME COMPLEXITY,O(1) SPACE COMPLEXITY

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

PROGRAM

```
#include<stdio.h>
int main()
{
    int n;
    scanf("%d",&n);

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

    for(int i=0;i<n;i++){

         for(int j=0;j<n;j++){

             if(i!=j){

                 if(array[j]-array[i]==d){

                     count=count+1;
                 }
             }
        }
    }

    if(count==0)
{
        printf("0");
    }
      else
            printf("1");
}
```

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! ✔