S.No: 1	Exp. Name: ***Write a C program to find the reverse of a given number***	Date:2023-04-01

Aim:

Design a C program which reverses the given number.

Source Code:

reverse.c

```
#include<stdio.h>
void main()
{
int n,r=0,rev=0;
scanf("%d",&n);
while(n>0)
{
r=n%10;
rev=rev*10+r;
n=n/10;
}
printf("Reversed number= %d",rev);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
456
Reversed number= 654

Test Case - 2
User Output
958745
Reversed number= 547859

ID: 224G1A0568

Srinivasa Ramanujan Institute of Technology

S.No: 2	Exp. Name: ***Write a C program to find second largest for the given numbers***	Date:2023-04-01

ID: 224G1A0568

Aim:

Design a C program which finds the `second maximum number` among the given one dimensional array of elements.

```
Sample Input and Output:Enter how many values you want to read : 6
Enter the value of a[0] : 45
Enter the value of a[1] : 24
Enter the value of a[2] : 23
Enter the value of a[3] : 65
Enter the value of a[4] : 78
Enter the value of a[5] : 42
The second largest element of the array = 65
```

Note: Do use the **printf()** function with a **newline** character (\n) at the end.

Source Code:

second_large.c

```
#include<stdio.h>
int main()
{
int i,n,a[20],max1=0,max2=0;
printf("Enter how many values you want to read : ");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("Enter the value of a[%d] : ",i);
scanf("%d",&a[i]);
}
for(i=0;i<n;i++)
{
if(max1<a[i])
{
max2=max1;
max1=a[i];
}
else if(a[i]>max2&&a[i]<max1)
{
max2=a[i];
}
}
printf("The second largest element of the array = %d\n",max2);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter how many values you want to read : 4
Enter the value of a[0] : 32

```
Enter the value of a[1] :  25
Enter the value of a[2] :  69
Enter the value of a[3] :  47
The second largest element of the array = 47
```


ID: 224G1A0568

S.No: 3	Exp. Name: ***Write a program which finds the kth smallest number among the given list of numbers.***	Date:2023-04-01

ID: 224G1A0568

Aim:

Write a program which finds the k^{th} smallest number among the given one dimensional array.

Sample Input and Ouput:

```
Enter how many values you want to read : 5
Enter the value of a[0] : 20
Enter the value of a[1] : 30
Enter the value of a[2] : 16
Enter the value of a[3] : 15
Enter the value of a[4] : 1
Enter which smallest element you want: 2
16 is the 2th smallest element
```

Hint: The k^{th} element refers to the index.

Source Code:

smallest.c

```
#include<stdio.h>
#define MAX 100
int main()
{
int a[MAX],i,n,j,kth,temp,pos;
printf("Enter how many values you want to read : ");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("Enter the value of a[%d] : ",i);
scanf("%d",&a[i]);
}
printf("Enter which smallest element you want: ");
scanf("%d",&kth);
for(i=0;i<n;i++)
{
pos=i;
for(j=i+1;j<n;j++)
if(a[j]<a[pos])
{
pos=j;
}
temp=a[i];
a[i]=a[pos];
a[pos]=temp;
}
printf("%d is the %dth smallest element",a[kth],kth);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter how many values you want to read : 5
Enter the value of a[0] : 20
Enter the value of a[1] : 30
Enter the value of a[2] : 16
Enter the value of a[3] : 15
Enter the value of a[4] : 1
Enter which smallest element you want: 2
16 is the 2th smallest element

Test Case - 2
User Output
Enter how many values you want to read : 6
Enter the value of a[0] : 32
Enter the value of a[1] : 65
Enter the value of a[2] : 98
Enter the value of a[3] : 74
Enter the value of a[4] : 12
Enter the value of a[5] : 15
Enter which smallest element you want: 4
74 is the 4th smallest element

ID: 224G1A0568

S.No: 4	Exp. Name: ***Design an algorithm and implement using C language the following exchanges***	Date:2023-04-01

ID: 224G1A0568

Aim:

Design an algorithm and implement using C language the following exchanges **a ← b ← c ← d ← a** and print the result as shown in the example.

```
Sample Input and Output:
Enter values of a, b, c and d: 98 74 21 36
After swapping
a = 74
b = 21
c = 36
d = 98
```

Source Code:

exchange.c

```
#include<stdio.h>
void main()
{
int a,b,c,d,temp;
printf("Enter values of a, b, c and d: ");
scanf("%d%d%d%d",&a,&b,&c,&d);
temp=a;
a=b;
b=c;
c=d;
d=temp;
printf("After swapping\na = %d\nb = %d\nc = %d\nd = %d\n",a,b,c,d);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter values of a, b, c and d: 1 2 3 4
After swapping
a = 2
b = 3
c = 4
d = 1

Test Case - 2
User Output
Enter values of a, b, c and d: 98 74 21 36
After swapping
a = 74
b = 21

c = 36
d = 98

S.No: 5	Exp. Name: ***Write a program to find the count of positive and negative numbers***	Date:2023-04-01

Aim:

Develop a C Program which counts the number of positive and negative numbers separately and also compute the sum of them.

```
Sample Input and Output:
How many numbers you want to add : 6
Enter number a[0] : 3
Enter number a[1] : 5
Enter number a[2] : -5
Enter number a[3] : 7
Enter number a[4] : -8
Enter number a[5] : 6
Count of positive numbers = 4
Sum of positive numbers = 21
Count of negative numbers = 2
Sum of Negative numbers = -13
```

Source Code:

count.c

```
#include<stdio.h>
int main()
{
int a[20],n,i,sump=0,sumn=0,countp=0,countn=0;
printf("How many numbers you want to add : ");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("Enter number a[%d] : ",i);
scanf("%d",&a[i]);
}
for(i=0;i<n;i++)
{
if(a[i]>0)
{
sump+=a[i];
countp=countp+1;
}
else
{
sumn+=a[i];
countn=countn+1;
}
}
printf("Count of positive numbers = %d\n",countp);
printf("Sum of positive numbers = %d\n",sump);
printf("Count of negative numbers = %d\n",countn);
printf("Sum of Negative numbers = %d\n",sumn);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
How many numbers you want to add : 5
Enter number a[0] : 4
Enter number a[1] : 5
Enter number a[2] : 6
Enter number a[3] : 2
Enter number a[4] : 6
Count of positive numbers = 5
Sum of positive numbers = 23
Count of negative numbers = 0
Sum of Negative numbers = 0

Test Case - 2
User Output
How many numbers you want to add : 4
Enter number a[0] : -4
Enter number a[1] : -1
Enter number a[2] : -3
Enter number a[3] : -2
Count of positive numbers = 0
Sum of positive numbers = 0
Count of negative numbers = 4
Sum of Negative numbers = -10

ID: 224G1A0568

S.No: 6	Exp. Name: ***Implement the C program which computes the sum of the first n terms of the series***	Date:2023-04-01

Aim:

Implement the C program which computes the sum of the first n terms of the series

Sum = 1 - 3 + 5 - 7 + 9 + ...

Sample Input and Output - 1:

```
Enter the value of n: 99
The sum of first 99 terms of the series is: 99
```

Source Code:

sum.c

```
#include<stdio.h>
void main()
{
int n,i,sum=0,sumn=0,sump=0;
printf("Enter the value of n: ");
scanf("%d",&n);
for(i=0;i<n;i++)
{
if(i%2==0)
{
sump+=2*i+1;
}
else
{
sumn+=-(2*i+1);
}
}
sum=sump+sumn;
printf("The sum of first %d terms of the series is: %d\n",n,sum);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the value of n: 789
The sum of first 789 terms of the series is: 789

Test Case - 2
User Output
Enter the value of n: 76
The sum of first 76 terms of the series is: -76

Test Case - 3

User Output
Enter the value of n: 99
The sum of first 99 terms of the series is: 99

S.No: 7	Exp. Name: ***Design a C program which determines factorial of numbers***	Date:2023-04-01

Aim:

Design a C program which determines the numbers whose factorial values are between(including) minimum and maximum values.

For example: The value of 6! is 720, 7! is 5040 and 8! is 40320. The factorial of 7 (5040) exists between the given limits.

Constraints: 1 <= min,max <= 103

Instruction: Your input and output layout must match exactly with the layout of the visible sample test cases.

Source Code:

factorial.c

```
#include<stdio.h>
void main()
{
int fact=1,i,max,min,x=1;
printf("Min: ");
scanf("%d",&min);
printf("Max: ");
scanf("%d",&max);
printf("Values: ");
for(i=1;i<=max;i++)
{
fact=fact*i;
if(fact>=min&&fact<=max)
{
if(x==1)
{
printf("%d ",i);
x=0;
}
else
printf("%d ",i);
}
}
printf("\n");
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Min: 5
Max: 10
Values: 3

Test Case - 2

User Output
Min: 5
Max: 29
Values: 3 4

S.No: 8	Exp. Name: ***Design an algorithm and implement using a C program which finds the sum of the infinite series***	Date:2023-04-01

ID: 224G1A0568

Aim:

Design an algorithm and implement using a C program which finds the `sum` of the `infinite series`
$1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots..,$
Print the result as shown in the example.

Sample Input and Output:

```
Enter the value of x and n: 4 5
sum = 3.666667
```

Source Code:

infinite.c

```
#include<stdio.h>
#include<math.h>
int main()
{
int x,n,m,i=0,fact=1;
float k,sum=0;
printf("Enter the value of x and n: ");
scanf("%d%d",&x,&n);
while(i<=n)
{
if(i%2==0)
{
fact=1;
for(m=1;m<=i;m++)
{
fact=fact*m;
}
k=(pow(x,i))/fact;
}
if(i%4!=0)
{
fact=1;
for(m=1;m<=i;m++)
{
fact=fact*m;
}
k=-(pow(x,i))/fact;
}
sum=sum+k;
i=i+2;
}
printf("sum = %f",sum);
}
```


Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the value of x and n: 4 5
sum = 3.666667

Test Case - 2
User Output
Enter the value of x and n: 12 5
sum = 793.000000

ID: 224G1A0568

Srinivasa Ramanujan Institute of Technology

S.No: 9	Exp. Name: ***Design a C program to print the sequence of numbers in which each number is the sum of the three most recent predecessors***	Date:2023-04-01

ID: 224G1A0568

Aim:

Design a C program to print the sequence of numbers in which each number is the sum of the three most recent predecessors. Assume first three numbers as **0**, **1**, and **1**, print the result as shown in the example.

Sample Input and Output:

```
Enter the number of terms: 7
First 7 terms in the series are:
0
1
1
2
4
7
13
```

Source Code:

first.c

```
#include<stdio.h>
int main()
{
int t1=0,t2=1,t3=1,t4,n,i;
printf("Enter the number of terms: ");
scanf("%d",&n);
printf("First %d terms in the series are:",n);
printf("\n%d\n%d\n%d\n",t1,t2,t3);
for(i=4;i<=n;i++)
{
t4=t1+t2+t3;
printf("%d\n",t4);
t1=t2;
t2=t3;
t3=t4;
}
return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the number of terms: 5
First 5 terms in the series are:
0
1
1
2

4

Test Case - 2
User Output
Enter the number of terms: 7
First 7 terms in the series are:
0
1
1
2
4
7
13

Test Case - 3
User Output
Enter the number of terms: 13
First 13 terms in the series are:
0
1
1
2
4
7
13
24
44
81
149
274
504

S.No: 10	Exp. Name: ***Write a C program to convert a Decimal number into binary, octal and hexadecimal number using a single user defined function.***	Date:2023-04-03

Aim:

Write a C program to convert a Decimal number into binary, octal and hexadecimal number using a single user defined function.

At the time of execution, the program should print the message on the console as:

Enter a positive decimal number :

```
For example, if the user gives the input as:
```

Enter a positive decimal number : 789

```
then the program should print the result as:
```

The binary number of decimal 789 is : 1100010101

```
The octal number of decimal 789 is : 1425
The hexadecimal number of decimal 789 is : 315
```

Note: Do use the **printf()** function with a **newline** character **(\n)** at the end.

Source Code:

oche.c

```
#include<stdio.h>
#include<math.h>
int main()
{
   int n,s,temp,bin[100],i,j;
   printf("Enter a positive decimal number : ");
   scanf("%d",&n);
   s=2*n;
   s=s/2;
   temp=s;
   for(i=0;s>0;i++)
   {
      bin[i]=s%2;
      s=s/2;
   }
   printf("The binary number of decimal %d is : ",temp);
   for(j=i-1;j>=0;j--)
   printf("%d",bin[j]);
   printf("\n");
   printf("The octal number of decimal %d is : %o\n",n,n);
   printf("The hexadecimal number of decimal %d is : %X\n",n,n);

}
```


Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter a positive decimal number : 45
The binary number of decimal 45 is : 101101
The octal number of decimal 45 is : 55
The hexadecimal number of decimal 45 is : 2D

Test Case - 2
User Output
Enter a positive decimal number : 10
The binary number of decimal 10 is : 1010
The octal number of decimal 10 is : 12
The hexadecimal number of decimal 10 is : A

Test Case - 3
User Output
Enter a positive decimal number : 6789
The binary number of decimal 6789 is : 1101010000101
The octal number of decimal 6789 is : 15205
The hexadecimal number of decimal 6789 is : 1A85

S.No: 11	Exp. Name: ***Develop an algorithm which computes the all the factors between 1 to 100 for a given number and implement it using C.***	Date:2023-04-01

ID: 224G1A0568

Aim:

Develop an algorithm which computes the all the factors between 1 to 100 for a given number and implement it using C.

Sample input output

Sample input output -1:

```
Enter a number: 23
Factors between 1 and 100 are: 1        23
```

Sample input output -2:

```
Enter a number: 234
Factors between 1 and 100 are: 1        2       3       6       9       13      18      26
```

Sample input output -3:

```
Enter a number: 5
Factors between 1 and 100 are: 1        5
```

Note: Do use the printf() function with a newline character (\n) at the end.

Source Code:

factors100.c

```
#include<stdio.h>
main()
{
   int i,n;
   printf("Enter a number: ");
   scanf("%d",&n);
   printf("Factors between 1 and 100 are: ");
   for(i=1;i<=100;i++)
   {
      if(n%i==0)
      printf("%d\t",i);
   }
   printf("\n");
   return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter a number: 45
Factors between 1 and 100 are: 1 3 5 9 15 45

S.No: 12	Exp. Name: ***Construct an algorithm which computes the sum of the factorials of numbers between m and n***	Date:2023-04-02

Aim:

Construct an algorithm which computes the sum of the factorials of numbers between m and n

Constraints:

m < n

Sample input output

Sample input output -1:

```
Enter m value: 3
Enter n value: 1
m value should be less than n
```

Sample input output -2:

```
Enter m value: 4
Enter n value: 6
Sum of factorials of numbers between 4 and 6 is 864
```

Sample input output -3:

```
Enter m value: 10
Enter n value: 13
Sum of factorials of numbers between 10 and 13 is 6749568000
```

Note: Do use the printf() function with a newline character (\n) at the end.

Note: Use an appropriate data type for the variable storing the sum to accommodate large factorial values.

Source Code:

fact.c

```
#include<stdio.h>
int main()
{
long int m,n,k,i,fact=1,sum=0;
printf("Enter m value: ");
scanf("%ld",&m);
printf("Enter n value: ");
scanf("%ld",&n);
if(m<n)
{
printf("Sum of factorials of numbers between %ld and %ld is ", m,n);
for(k=m;k<=n;k++)
{
fact = 1;
for(i=k;i>=1;i--)
{
fact=fact*i;
```

```
}
sum=sum+fact;
}
printf("%ld\n",sum);
}
else
printf("m value should be less than n\n");
return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter m value: 10
Enter n value: 13
Sum of factorials of numbers between 10 and 13 is 6749568000

Test Case - 2
User Output
Enter m value: 3
Enter n value: 1
m value should be less than n

ID: 224G1A0568

S.No: 13	Exp. Name: ***Write a C program to display the elements of an array in reverse order***	Date:2023-04-02

Aim:

Write a program to **print** the given integer elements of an array (with max size 10) in reverse order.

At the time of execution, the program should print the message on the console as:
Enter size of the array :
For example, if the user gives the **input** as:

```
Enter size of the array : 3
```

Next, the program should **print** the message on the console as:

```
Enter array elements :
```

If the user gives the **input** as:

```
Enter array elements : 10 20 30
```

then the program should **print** the result as:

```
Array elements in reverse order : 30 20 10
```

[**Hint:** First read an integers from standard input into the array and then use a loop to iterate on that array in the reverse order (meaning starting from the last element till the first) to print the elements.]

Note: Do use the printf() function without a newline character (\n).

Source Code:

print.c

```
#include<stdio.h>
main()
{
   int k,a[100],n,b;
   printf("Enter size of the array : ");
   scanf("%d",&n);
   int size = a[n];
   printf("Enter array elements : ");
   for(k=0;k<n;k++)
   {
      scanf("%d",&a[k]);
   }
   printf("Array elements in reverse order : ");
   for(k=n-1;k>=0;k--)
   {
      printf("%d ",a[k]);
   }
   printf("\n");
   return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter size of the array : 3
Enter array elements : 10 20 30
Array elements in reverse order : 30 20 10

Test Case - 2
User Output
Enter size of the array : 6
Enter array elements : 11 88 66 22 33 44
Array elements in reverse order : 44 33 22 66 88 11

ID: 224G1A0568

S.No: 14	Exp. Name: ***Program - Addition of two matrices***	Date:2023-04-02

Aim:

The below sample code finds the **addition** of two matrices.

In the **main()** function read a two two-dimensional array of elements and then find the **addition** of two matrices.

The **logic** is
First checks the **row sizes** and **column sizes** of two two-dimensional arrays are equal or not.
If the sizes are not equal then print "Addition is not possible" and stop the process.
If the sizes are equal then use **two for loops** to add each corresponding elements of two matrices and finally print the result.
Fill in the missing code so that it produces the desired output.

Source Code:

matrix.c

```
#include<stdio.h>
void main()
{
   int i,j,m,n,p,q;
   int a[10][10],b[10][10],c[10][10];
   printf("Enter the row & column sizes of matrix-1 : ");
   scanf("%d %d",&m,&n);
   printf("Enter matrix-1 %d elements : ",m*n);
   for(i=0;i<m;i++)
   {
      for(j=0;j<n;j++)
      {
         scanf("%d",&a[i][j]);
      }
   }
   printf("Enter the row & column sizes of matrix-2 : ");
   scanf("%d %d",&p,&q);
   printf("Enter matrix-2 %d elements : ",p*q);
   for(i=0;i<p;i++)
   {
      for(j=0;j<q;j++)
      {
         scanf("%d",&b[i][j]);
      }
   }
   printf("The given matrix-1 is\n");
   for(i=0;i<m;i++)
   {
      for(j=0;j<n;j++)
      {
         printf("%d ",a[i][j]);
      }
      printf("\n");
   }
   printf("The given matrix-2 is\n");
   for(i=0;i<p;i++)
```

```
    {
       for(j=0;j<q;j++)
       {
          printf("%d ",b[i][j]);
       }
       printf("\n");
    }
    if(m==p&&n==q)
    {
       for(i=0;i<m;i++)
       {
       for(j=0;j<n;j++)
       {
          c[i][j]=a[i][j]+b[i][j];
       }
       }
       printf("Addition of two matrices is\n");
       for(i=0;i<m;i++)
       {
          for(j=0;j<n;j++)
          {
             printf("%d ",c[i][j]);
          }
          printf("\n");
       }
       }
       else
       {
          printf("Addition is not possible\n");
       }
    }
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the row & column sizes of matrix-1 : 2 2
Enter matrix-1 4 elements : 1 2 3 4
Enter the row & column sizes of matrix-2 : 2 2
Enter matrix-2 4 elements : 4 5 6 7
The given matrix-1 is
1 2
3 4
The given matrix-2 is
4 5
6 7
Addition of two matrices is
5 7
9 11

S.No: 15	Exp. Name: ***Program - Subtraction of two matrices***	Date:2023-04-02

Aim:

The below sample code finds the **subtraction** of two matrices.

In the **main()** function read a two two-dimensional array of elements and then find the **subtraction** of two matrices.

The **logic** is
First checks the **row sizes** and **column sizes** of two two-dimensional arrays are equal or not.
If the sizes are not equal then print "subtraction is not possible" and stop the process.
If the sizes are equal then use **two for loops** to subtract each corresponding elements of two matrices and finally print the result.
Fill in the missing code so that it produces the desired output.

Source Code:

submatrix.c

```
#include<stdio.h>
void main()
{
   int i,j,p,q,r,s;
   int a[5][5],b[5][5],c[5][5];
   printf("Enter the row & column sizes of matrix-1 : ");
   scanf("%d%d",&p,&q);
   printf("Enter matrix-1 %d elements : ",p*q);
   for(i=0;i<p;i++)
   {
      for(j=0;j<q;j++)
      {
         scanf("%d",&a[i][j]);
      }
   }
   printf("Enter the row & column sizes of matrix-2 : ");
   scanf("%d%d",&r,&s);
   printf("Enter matrix-2 %d elements : ",r*s);
   for(i=0;i<r;i++)
   {
      for(j=0;j<s;j++)
      {
         scanf("%d",&b[i][j]);
      }
   }
   printf("The given matrix-1 is\n");
   for(i=0;i<p;i++)
   {
      for(j=0;j<q;j++)
      {
         printf("%d ",a[i][j]);
      }
      printf("\n");
   }
   printf("The given matrix-2 is\n");
   for(i=0;i<r;i++)
```

```
    {
       for(j=0;j<s;j++)
       {
          printf("%d ",b[i][j]);
       }
       printf("\n");
    }
    if(p==r && q==s)
    {
       for(i=0;i<p;i++)
       {
          for(j=0;j<q;j++)
          {
             c[i][j]=a[i][j]-b[i][j];
          }
       }
       printf("Subtraction of two matrices is\n");
       for(i=0;i<p;i++)
       {
          for(j=0;j<q;j++)
       {
          printf("%d ",c[i][j]);
       }
       printf("\n");
    }
}
else
{
   printf("Subtraction is not possible\n");
}
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the row & column sizes of matrix-1 : 2 2
Enter matrix-1 4 elements : 6 4 8 1
Enter the row & column sizes of matrix-2 : 2 2
Enter matrix-2 4 elements : 1 2 3 4
The given matrix-1 is
6 4
8 1
The given matrix-2 is
1 2
3 4
Subtraction of two matrices is
5 2
5 -3

S.No: 16	Exp. Name: ***Write a C program to perform matrix multiplication***	Date:2023-04-02

ID: 224G1A0568

Aim:

Write a C program to perform matrix multiplication on two dimensional matrix.

At the time of execution, the program should print the message on the console as:

```
Enter the row & column sizes of matrix-1 :
```

For example, if the user gives the input as:

```
Enter the row & column sizes of matrix-1 : 2 2
```

Next, the program should print the message on the console as:

```
Enter matrix-1 4 elements :
```

If the user gives the input as:

```
Enter matrix-1 4 elements : 1 1 2 2
```

Next, the program should print the message on the console as:

```
Enter the row & column sizes of matrix-2 :
```

If the user gives the input as:

```
Enter the row & column sizes of matrix-2 : 2 2
```

Next, the program should print the message on the console as:

```
Enter matrix-2 4 elements :
```

If the user gives the input as:

```
Enter matrix-2 4 elements : 1 2 7 4
```

Then the program should print the result as:

```
The given matrix-1 is
1 1
2 2
The given matrix-2 is
1 2
7 4
Multiplication of two matrices is
8 6
16 12
```

Otherwise, the program should print the result as :

```
Multiplication is not possible
```

Note: Do use the printf() function with a newline character(\n).

Source Code:

matmul.c

```
#include<stdio.h>
void main()
{
   int i,j,k,m,n,p,q;
   int a[5][5],b[5][5],c[5][5];
   printf("Enter the row & column sizes of matrix-1 : ");
   scanf("%d %d",&m,&n);
   printf("Enter matrix-1 %d elements : ",m*n);
   for(i=0;i<m;i++)
   {
      for(j=0;j<n;j++)
      {
         scanf("%d",&a[i][j]);
      }
   }
   printf("Enter the row & column sizes of matrix-2 : ");
   scanf("%d %d",&p,&q);
   printf("Enter matrix-2 %d elements : ",p*q);
   for(i=0;i<p;i++)
   {
      for(j=0;j<q;j++)
      {
         scanf("%d",&b[i][j]);
      }
   }
   printf("The given matrix-1 is\n");
   for(i=0;i<m;i++)
   {
      for(j=0;j<n;j++)
      {
         printf("%d ",a[i][j]);
      }
      printf("\n");
   }
      printf("The given matrix-2 is\n");
      for(i=0;i<p;i++)
      {
         for(j=0;j<q;j++)
         {
            printf("%d ",b[i][j]);
         }
         printf("\n");
      }
      if(n==p)
      {
```


ID: 224G1A0568
2022-2026-CSE-A
Srinivasa Ramanujan Institute of Technology

```
         for(i=0;i<m;i++)
         {
            for(j=0;j<q;j++)
            {
               c[i][j]=0;
               for(k=0;k<p;k++)
               {
                  c[i][j]=c[i][j]+a[i][k]*b[k][j];
               }
            }
         }
         printf("Multiplication of two matrices is\n");
         for(i=0;i<m;i++)
         {
         for(j=0;j<q;j++)
         {
            printf("%d ",c[i][j]);
         }
         printf("\n");
      }
   }
   else
   {
      printf("Multiplication is not possible\n");
   }
   }
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the row & column sizes of matrix-1 : 2 2
Enter matrix-1 4 elements : 1 2 3 4
Enter the row & column sizes of matrix-2 : 2 2
Enter matrix-2 4 elements : 4 5 6 7
The given matrix-1 is
1 2
3 4
The given matrix-2 is
4 5
6 7
Multiplication of two matrices is
16 19
36 43

Test Case - 2
User Output
Enter the row & column sizes of matrix-1 : 2 2
Enter matrix-1 4 elements : 1 1 2 2
Enter the row & column sizes of matrix-2 : 2 2
Enter matrix-2 4 elements : 1 2 7 4
The given matrix-1 is
1 1

ID: 224G1A0568

2 2
The given matrix-2 is
1 2
7 4
Multiplication of two matrices is
8 6
16 12

S.No: 17	Exp. Name: ***Write a C program to implement the string manipulation operations by using library functions.***	Date:2023-04-02

ID: 224G1A0568

Aim:

Write a program to implement the string manipulation operations by using string library functions.

At the time of execution, the program should print the message on the console as:

```
Enter two strings :
```

For example, if the user gives the input as:

```
Enter two strings : Ram Laxman
```

then the program should print the result as:

```
The length of Ram : 3
The copied string of Ram : Ram
Ram is greater than Laxman
The concatenated string : RamLaxman
```

Note: Do use the printf() function with a newline character (\n) at the end.

Source Code:

str.c

```
#include<stdio.h>
#include<string.h>
void main()
{
   char str1[100], str2[100];
   int len;
   printf("Enter two strings : ");
   scanf("%s %s",str1,str2);
   len= strlen(str1);
   printf("The length of %s : %d\n",str1,len);
   printf("The copied string of %s : %s\n",str1,strcpy(str1,str1));
   int i=strcmp(str1,str2);
   if(i==0)
   {
      printf("Both strings are equal\n",str1,str2);
   }
   else if(i>0)
   {
      printf("%s is greater than %s\n",str1,str2);
   }
   else
   {
      printf("%s is less than %s\n",str1,str2);
   }
   printf("The concatenated string : %s\n",strcat(str1,str2));
   printf("\n");
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter two strings : Ram Laxman
The length of Ram : 3
The copied string of Ram : Ram
Ram is greater than Laxman
The concatenated string : RamLaxman

Test Case - 2
User Output
Enter two strings : Faculty Bird
The length of Faculty : 7
The copied string of Faculty : Faculty
Faculty is greater than Bird
The concatenated string : FacultyBird

ID: 224G1A0568
2022-2026-CSE-A
Srinivasa Ramanujan Institute of Technology

S.No: 18	Exp. Name: ***given a list of n numbers, Design an algorithm which prints the number of stars equivalent to the value of the number. The stars for each number should be printed horizontally.***	Date:2023-04-02

ID: 224G1A0568

2022-2026-CSE-A

Srinivasa Ramanujan Institute of Technology

Aim:

Take a list of n numbers, Design an algorithm which prints the number of stars equivalent to the value of the number. The stars for each number should be printed horizontally.

Sample input output

```
Sample input output -1:

Enter the number of numbers: 6
Enter number 1: 4
Enter number 2: 6
Enter number 3: 9
Enter number 4: 5
Enter number 5: 2
Enter number 6: 6
****
******
*********
*****
**
******

Sample input output -2:

Enter the number of numbers: 4
Enter number 1: 4
Enter number 2: 2
Enter number 3: 1
Enter number 4: 3
****
**
*
***
```

Note: Do use the printf() function with a newline character (\n) at the end.

Source Code:

star.c

```
#include<stdio.h>
int main()
{
   int n,j,i,a[10];
   printf("Enter the number of numbers: ");
   scanf("%d",&n);
   for(i=0;i<n;i++)
   {
      printf("Enter number %d: ",i+1);
      scanf("%d",&a[i]);
   }
```

```
    for(i=0;i<n;i++)
    {
       for(j=1;j<=a[i];j++)
       {
          printf("*");
       }
       printf("\n");
    }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the number of numbers: 6
Enter number 1: 4
Enter number 2: 6
Enter number 3: 9
Enter number 4: 5
Enter number 5: 2
Enter number 6: 6
****
******
****
*****
**
******

Test Case - 2
User Output
Enter the number of numbers: 5
Enter number 1: 5
Enter number 2: 4
Enter number 3: 3
Enter number 4: 2
Enter number 5: 1
*****
****
***
**
*

ID: 224G1A0568
2022-2026-CSE-A
Srinivasa Ramanujan Institute of Technology

S.No: 19	Exp. Name: ***Write a C program to sort elements using insertion sort***	Date:2023-04-02

Aim:

Write a program to sort the elements in ascending order with insertion sort technique using functions.

At the time of execution, the program should print the message on the console as:

```
Enter n value :
```

For example, if the user gives the input as:

```
Enter n value : 3
```

Next, the program should print the message on the console as:

```
Enter 3 elements :
```

if the user gives the input as:

```
Enter 3 elements : 45 67 34
```

then the program should print the result as:

```
Elements before sorting : 45 67 34
Elements after sorting : 34 45 67
```

Note: Do use printf() with '\n' at the end of output.

Source Code:

sort.c

```
#include<stdio.h>
void insertion_sort(int [], int);
void read(int [], int);
void display(int [], int);
void main()
{
   int a[20],n,i;
   printf("Enter n value : ");
   scanf("%d",&n);
   read(a,n);
   printf("Elements before sorting : ");
   display(a,n);
   insertion_sort(a,n);
   printf("Elements after sorting : ");
   display(a,n);
}
void insertion_sort(int a[],int n)
{
   int i,j,k;
   for(i=1;i<n;i++)
   {
      k=a[i];
      j=i-1;
      while(j>=0&&a[j]>k)
      {
```

```
         a[j+1]=a[j];
         j=j-1;
   }
   a[j+1]=k;
}
}
void read(int a[],int n)
{
   int i;
   printf("Enter %d elements : ",n);
   for(i=0;i<n;i++)
   scanf("%d",&a[i]);
}
void display(int a[],int n)
{
   int i;
   for(i=0;i<n;i++)
   printf("%d ",a[i]);
   printf("\n");
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter n value : 3
Enter 3 elements : 45 67 34
Elements before sorting : 45 67 34
Elements after sorting : 34 45 67

ID: 224G1A0568

S.No: 20	Exp. Name: ***Write a C program to sort the list of numbers using bubble sort and functions***	Date:2023-04-02

ID: 224G1A0568

Aim:

Write a program to sort the elements in descending order with bubble sort technique using functions.

At the time of execution, the program should print the message on the console as:

```
Enter n value :
```

For example, if the user gives the input as:

```
Enter n value : 3
```

Next, the program should print the message on the console as:

```
Enter 3 elements :
```

if the user gives the input as:

```
Enter 3 elements : 45 67 34
```

then the program should print the result as:

```
Elements before sorting : 45 67 34
Elements after sorting : 67 45 34
```

Note: Write the functions read(), bubbleSort() and display() in sorta.c.

Source Code:

sort.c

```
#include <stdio.h>
void bubbleSort(int [],int);
void read(int [],int);
void display(int[],int);
void main()
{
   int a[20], n, i;
   printf("Enter n value : ");
   scanf("%d",&n);
   read(a,n);
   printf("Elements before sorting : ");
   display(a, n);
   bubbleSort(a, n);
   printf("Elements after sorting : ");
   display(a, n);
}
void read(int a[],int n)
{
   int i;
   printf("Enter %d elements : ",n);
for(i=0;i<n;i++)
scanf("%d",&a[i]);
}
void display(int a[],int n)
```

```
{
   int i;
   for(i=0;i<n;i++)
   printf("%d ",a[i]);
   printf("\n");
}
void bubbleSort(int a[],int n)
{
   int i,j,temp;
   for(i=0;i<n-1;i++)
   {
      for(j=i+1;j<n;j++)
      {
         if(a[j]>a[i])
         {
            temp=a[i];
            a[i]=a[j];
            a[j]=temp;
         }
      }
   }
}
```

sorta.c

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

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter n value : 3
Enter 3 elements : 4 6 8
Elements before sorting : 4 6 8
Elements after sorting : 8 6 4

Test Case - 2
User Output
Enter n value : 5
Enter 5 elements : 34 56 71 26 17
Elements before sorting : 34 56 71 26 17
Elements after sorting : 71 56 34 26 17

S.No: 21	Exp. Name: ***Write a C program to sort the elements using selection sort - largest element method technique***	Date:2023-04-02

ID: 224G1A0568

Aim:

Write a program to sort the given array elements using selection sort largest element method.

At the time of execution, the program should print the message on the console as:

```
Enter value of n :
```

For example, if the user gives the input as:

```
Enter value of n : 3
```

Next, the program should print the messages one by one on the console as:

```
Enter element for a[0] :
Enter element for a[1] :
Enter element for a[2] :
```

if the user gives the input as:

```
Enter element for a[0] : 22
Enter element for a[1] : 33
Enter element for a[2] : 12
```

then the program should print the result as:

```
Before sorting the elements in the array are
Value of a[0] = 22
Value of a[1] = 33
Value of a[2] = 12
After sorting the elements in the array are
Value of a[0] = 12
Value of a[1] = 22
Value of a[2] = 33
```

Fill in the missing code so that it produces the desired result.

Source Code:

array.c

```
#include<stdio.h>
void main()
{
  int a[20],i,n,j,max,temp=0;
  printf("Enter value of n : ");
  scanf("%d",&n);
  for(i=0;i<n;i++)
  {
     printf("Enter element for a[%d] : ",i);
     scanf("%d",&a[i]);
  }
  printf("Before sorting the elements in the array are\n");
  for(i=0;i<n;i++)
  printf("Value of a[%d] = %d\n",i,a[i]);
```

```
   for(i=n-1;i>0;i--)
   {
      max=j;
      for(j=1;j>=0;j--)
      {
         if(a[j]>=a[max])
         max=j;
      }
      temp=a[i];
      a[i]=a[max];
      a[max]=temp;
   }
   printf("After sorting the elements in the array are\n");
   for(i=0;i<n;i++)
   {
      printf("Value of a[%d] = %d\n",i,a[i]);
   }
 }
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter value of n : 3
Enter element for a[0] : 15 68 48
Enter element for a[1] : Enter element for a[2] : Before sorting the elements in th
Value of a[0] = 15
Value of a[1] = 68
Value of a[2] = 48
After sorting the elements in the array are
Value of a[0] = 15
Value of a[1] = 48
Value of a[2] = 68

ID: 224G1A0568

S.No: 22	Exp. Name: ***Write a C program to sort given elements using quick sort***	Date:2023-04-03

ID: 224G1A0568

Aim:

Write a program to sort (Ascending order) the given elements using quick sort technique.

Note: Pick the first element as pivot. You will not be awarded marks if you do not follow this instruction.

At the time of execution, the program should print the message on the console as:

```
Enter array size :
```

For example, if the user gives the input as:

```
Enter array size : 5
```

Next, the program should print the following message on the console as:

```
Enter 5 elements :
```

if the user gives the input as:

```
Enter 5 elements : 34 67 12 45 22
```

then the program should print the result as:

```
Before sorting the elements are : 34 67 12 45 22
After sorting the elements are : 12 22 34 45 67
```

Note: Do use the **printf()** function with a newline character **(\n).**

Source Code:

quicksortmain.c

```
#include<stdio.h>
#include "quicksortfunctions.c"
int main()
{
   int arr[20];
   int i,n;
   printf("Enter array size : ");
   scanf("%d",&n);
   printf("Enter %d elements : ",n);
   for(i=0;i<n;i++)
   {
      scanf("%d",&arr[i]);
   }
   printf("Before sorting the elements are : ");
   for(i=0;i<n;i++)
   {
      printf("%d ",arr[i]);
   }
   quicksort(arr,0,n-1);
   printf("\nAfter sorting the elements are : ");
   for(i=0;i<n;i++)
{
   printf("%d ",arr[i]);
```

```
}
   printf("\n");
}
```

quicksortfunctions.c

```
int temp,pivolt,left,right;
void quicksort(int a[20],int low,int high)
{
   left=low;
   right=high;
   pivolt=a[(low+high)/2];
   do
   {
      while(a[left]<pivolt)
      left++;
      while(a[right]>pivolt)
      right--;
      if(left<=right)
      {
         temp=a[left];
         a[left]=a[right];
         a[right]=temp;
         right--;
         left++;
      }
   }
   while(left<=right);
   if(low<right)
   quicksort(a,low,right);
   if(left<high)
   quicksort(a,left,high);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter array size : 5
Enter 5 elements : 34 67 12 45 22
Before sorting the elements are : 34 67 12 45 22
After sorting the elements are : 12 22 34 45 67

Test Case - 2
User Output
Enter array size : 8
Enter 8 elements : 77 55 22 44 99 33 11 66
Before sorting the elements are : 77 55 22 44 99 33 11 66
After sorting the elements are : 11 22 33 44 55 66 77 99

Test Case - 3

ID: 224G1A0568

User Output
Enter array size : 5
Enter 5 elements : -32 -45 -67 -46 -14
Before sorting the elements are : -32 -45 -67 -46 -14
After sorting the elements are : -67 -46 -45 -32 -14

S.No: 23	Exp. Name: ***Illustrate the use of auto variable***	Date:2023-04-02

ID: 224G1A0568

Aim:

Illustrate the use of auto variable.
The variables defined using **auto** storage class are called as local variables.
Auto stands for **automatic** storage class. A variable is in auto storage class by default if it is not explicitly specified.
The scope of an auto variable is **limited with the particular block only.**
Once the control goes out of the block, the access is destroyed. This means only the block in which the auto variable is declared can access it.
A keyword **auto** is used to define an auto storage class. By default, an auto variable contains a **garbage value**.
Follow the instructions given in the comment lines to declare auto variables and print their values at different places in the program.

Source Code:

auto.c

```
#include<stdio.h>
void main() {
   auto int d=10;
   // Declare an auto variable d of type integer.
   // Print the value of d.
   {
      auto int d=4;
       // Declare and initialize the auto variable d with 4.
      {
         auto int d=6;
         printf("d=%d\n",d);
         // Declare and initialize the auto variable d with 6/
         // Print the value of d.
      }
      printf("d=%d\n",d);
      // Print the value of d.
   }
   printf("d=%d\n",d);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
32767
6
4

S.No: 24	Exp. Name: ***Illustrate the use of static variables***	Date:2023-04-03

ID: 224G1A0568

Aim:

Illustrate the use of static variables
The **static** variables are used within function/ file as local static variables.
They can also be used as a global variable
Static local variable is a local variable that retains and stores its value between function calls or block and remains visible only to the function or block in which it is defined.
Static global variables are global variables visible only to the file in which it is declared.
Static variable has a default initial value zero and is initialized only once in its lifetime.
Follow the instructions given in the comment lines to declare and initialize the static variables and understand the working of static variables.

Source Code:

static.c

```
#include <stdio.h>
void next(void);
static int counter=5;
// Declare a global static variable 'counter' with an initial value of 5.
void main() {
   while(counter<10) {
      next();
      counter++;
   }
   return 0;
}
void next( void ) {
   static int iteration=10;
   // Declare a static integer variable 'iteration' with an initial value 10
   iteration ++;
   printf("iteration=%d and counter= %d\n", iteration, counter);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
iteration=11 and counter= 5
iteration=12 and counter= 6
iteration=13 and counter= 7
iteration=14 and counter= 8
iteration=15 and counter= 9

S.No: 25	Exp. Name: ***Illustrate the use of register variables***	Date:2023-04-02

Aim:

Illustrate the use of register variables.

- You can use the **register** storage class when you want to store local variables within functions or blocks in CPU registers instead of RAM to have quick access to these variables. For example, "counters" are a good candidate to be stored in the register.
- The keyword **register** is used to declare a register storage class. The variables declared using register storage class has lifespan throughout the program.
- It is similar to the auto storage class. The variable is limited to the particular block. The only difference is that the variables declared using register storage class are stored inside CPU registers instead of a memory. Register has faster access than that of the main memory.
- The variables declared using register storage class has no default value. These variables are often declared at the beginning of a program.
- Accessing the address of the register variables results in an error.

Try it out

```
A statement like
int *ptr = &weight;
will result in an error like
int *ptr = &weight;
address of register variable 'weight' requested
```

Follow the instructions given in the comment lines to understand the working of register variables.

Source Code:

register.c

```
#include <stdio.h>
void main() {
   register int weight;
   // Declare a register variable weight of type int.
   printf("The default weight value is: %d\n",weight);
   weight=65;
   printf("The current weight value is : %d\n",weight);
   // Add the line described above to obtain the error.
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
The default weight value is: 1024482696

S.No: 26	Exp. Name: ***Illustrate the use of extern variables***	Date:2023-04-02

Aim:

Illustrate the use of extern variables.

Follow the instructions given in the comment lines to write code and the working of the extern variables.

Source Code:

main.c

```
// Use the variable initialized in extrafile.c
#include"extrafile.c"
void main() {
   printf("Value of the external integer is = %d\n", i);
}
```

extrafile.c

```
#include <stdio.h>
int i=51;
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Value of the external integer is = 51

ID: 224G1A0568

Srinivasa Ramanujan Institute of Technology

S.No: 27	Exp. Name: ***Develop a C program which takes two numbers as command line arguments and finds all the common factors of those two numbers.***	Date:2023-04-02

Aim:

Develop a C program which takes two numbers as command line arguments and finds all the common factors of those two numbers.

Sample input output

```
Sample input output -1:

Cmd Args : 10 20
Common factors for 10 and 20 are: 1     2       5       10

Sample input output -2:

Cmd Args : 45 23
Common factors for 45 and 23 are: 1
```

Note: Do use the printf() function with a newline character (\n) at the end.

Source Code:

common_factors.c

```
#include<stdio.h>
#include<stdlib.h>
int main(int argc,char*argv[])
{
   int a,b;
   int i,small;
   a=atoi(argv[1]);
   b=atoi(argv[2]);
   small=(a<b)?a:b;
   printf("Common factors for %d and %d are: ",a,b);
   for(i=1;i<=small;i++)
   {
      if(a%i==0&&b%i==0)
      printf("%d\t",i);
   }
   printf("\n");
   return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Common factors for 10 and 20 are: 1 2 5 10

Test Case - 2
User Output

```
Common factors for 18 and 39 are: 1     3
```

S.No: 28	Exp. Name: ***Design a C program that sorts the strings using array of pointers.***	Date:2023-04-02

Aim:

Design a C program that sorts the strings using array of pointers.

Sample input output

```
Sample input-output -1:

Enter the number of strings: 2
Enter string 1: Tantra
Enter string 2: Code
Before Sorting
Tantra
Code
After Sorting
Code
Tantra
Sample input-output -2:

Enter the number of strings: 3
Enter string 1: India
Enter string 2: USA
Enter string 3: Japan
Before Sorting
India
USA
Japan
After Sorting
India
Japan
USA
```

Source Code:

stringssort.c

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
void main()
{
   char * temp;
   int i,j,diff,num_strings;
   char * strArray[10];
   printf("Enter the number of strings: ");
   scanf("%d",&num_strings);
   if(num_strings>10)
   {
      printf("Sorry,maximum strings allowed is %d. Defaulting.",10);
      num_strings =10;
   }
   for(i=0;i<num_strings;i++)
   {
```

```
      printf("Enter string %d: ",i+1);
      strArray[i] =(char *) malloc(10 *sizeof(char));
      scanf("%s",strArray[i]);
   }
   printf("Before Sorting\n");
   for(i=0;i<num_strings;i++)
   {
      printf("%s\n",strArray[i]);
   }
   sort(strArray,num_strings);
   printf("After Sorting\n");
   for(i=0; i < num_strings ;i++){
      printf("%s\n",strArray[i]);
   }
}
void sort(char *s[],int num_strings)
{
   char*temp;
   int item,i;
   for(item=0; item < num_strings; item++)
   {
      temp=s[item];
      for(i=item;i > 0 && strcasecmp(s[i -1],temp)>0;i--);
   {
         memmove(&s[i + 1],&s[i],(item-i) * sizeof(char *));
         s[i] =temp;
   }
}
return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the number of strings: 2
Enter string 1: Tantra
Enter string 2: Code
Before Sorting
Tantra
Code
After Sorting
Code
Tantra

Test Case - 2
User Output
Enter the number of strings: 3
Enter string 1: Dhoni
Enter string 2: Kohli
Enter string 3: Rohit
Before Sorting
Dhoni

ID: 224G1A0568
2022-2026-CSE-A
Srinivasa Ramanujan Institute of Technology

Kohli
Rohit
After Sorting
Dhoni
Kohli
Rohit