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

ID: 224G1A0534

Srinivasa Ramanujan Institute of Technology

Aim:

Design a C program which reverses the given number.

Source Code:

reverse.c

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

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

ID: 224G1A0534

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);
}
```

2022-2026-CSE-A

Srinivasa Ramanujan Institute of Technology

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: 224G1A0534

2022-2026-CSE-A

Srinivasa Ramanujan Institute of Technology

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

Aim:

Write a program which finds thek^{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: Thek^{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

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

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>
int 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:

```
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: 2023-04-08

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

S.No: 6	Exp. Name: ***Implement the C program which computes the sum of the first n terms of the series***	Date: 2023-04-08

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: 8	Exp. Name: ***Design an algorithm and implement using a C program which finds the sum of the infinite series***	Date: 2023-04-06

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

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: 2023-04-08

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: 2023-04-13

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: 2023-04-06

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  39  78
```

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>
int 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: 2023-04-09**

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("%d",&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

S.No: 13	Exp. Name: ***Write a C program to display the elements of an array in reverse order***	Date: 2023-04-06

ID: 224G1A0534

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>
 int 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;
}
```

2022-2026-CSE-A

Srinivasa Ramanujan Institute of Technology

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

S.No: 17	Exp. Name: ***Write a C program to implement the string manipulation operations by using library functions.***	**Date: 2023-04-06**

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

S.No: 23	Exp. Name: ***Illustrate the use of auto variable***	Date: 2023-04-03

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() {
        // Declare an auto variable d of type integer.
        // Print the value of d.
        {
                 // Declare and initialize the auto variable d with 4.
                {
                        // Declare and initialize the auto variable d with 6/
                        // Print the value of d.
                }
                // Print the value of 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: 2023-04-14

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;
void main()
{
         while(counter<10)
         {
                    next();
                    counter++;
         }
         return 0;
}
void next( void)
{
        static int iteration=10;
        iteration ++;
        printf("iteration=%d and couter %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: 2023-04-06

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 variabl weight of type int.
printf("The default weight value is: %d\n",weight);
weight=65;
printf("The current weight value is: %\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: 2023-04-03

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: 224G1A0534

Srinivasa Ramanujan Institute of Technology