Practical 21

Aim:

To create an array of 10 elements and to show the sum and average of 10 elements entered.

Description:

The user is first asked to enter the size of the array i.e., n. After the size is asked from the user, we ask the user to enter the elements of the array. 'For' loop is iterated for the inputs of the array elements and then we print the elements the user entered. Again a 'for' loop is iterated for "sum" and "average" and the result is printed.

Program:

```
#include<stdio.h>
int main()
{
    int i, n, a[10], sum=0, avg; printf("Enter
    the size of the array: \n"); scanf("%d",
    &n);
    printf("Enter the array elements:\n");
    for(i=0;i<n;i++)
    {
        scanf("%d", &a[i]);
    }
    printf("The array elements are:\n");
    for(i=0;i<n;i++)
    {
        printf("%d\n", a[i]);
    }
    for(i=0;i<n;i++)
    {
```

```
        sum=sum+a[i];
    }
    printf("Sum: %d\n", sum);
    for(i=0;i<n;i++)
    {
        avg=sum/n;
    }
    printf("Avg: %d\n", avg);
    return 0;

}
```

Output:

```
Enter the size of the array:
5
Enter the array elements:
1
2
3
4
5
The array elements are:
1
2
3
4
5
Sum:  15
Avg:  3

...Program finished with exit code 0
Press ENTER to exit console.
```

Practical 22

Aim:
To find the maximum number in the given array.

Description:
First take the size from the user and ask the user to enter the elements. To find the largest element, the first two elements of the array are checked and the largest of these two elements are placed in a[0]. The first and third elements are checked and the largest of these two elements is placed in a[0]. This process continues until the first and last elements are checked. Then the largest number will be stored in the a[0] position (and max=a[0]).

Program:

```
#include<stdio.h>
int main()
{
    int i, n, a[10], max;
    printf("Enter the size of the array: \n");
    scanf("%d", &n);
    printf("Enter the array elements:\n");
    for(i=0;i<n;i++)
    {
        scanf("%d", &a[i]);
    }
    max=a[0];
    for(i=0;i<n;i++)
    {
        if (max<a[i])
        max=a[i];
```

```
    }
    printf("The max is:%d\n", max);
    return 0;
}
```

Output:

```
Enter the size of the array:
8
Enter the array elements:
33
24
55
78
11
566
33
2345
The max is:2345


...Program finished with exit code 0
Press ENTER to exit console.
```

Practical 23

Aim:
To display a matrix.

Description:
As a matrix is a 2-D array, we will need 2 'for' loops; one for variable 'i' and another for variable 'j'. And the same logic as the 1-D array.

Program:

```
#include<stdio.h>
int main()
{
    int i, j, a[10][10];
    printf("Enter the array elements:\n");
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            scanf("%d", &a[i][j]);
        }
    }
    printf("The matrix is:\n");
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            printf("%d\t", a[i][j]);

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

```
    return 0;
}
```

Output:

```
Enter the array elements:
1
2
3
4
5
6
78
90
1
The matrix is:
1          2          3
4          5          6
78         90         1


...Program finished with exit code 0
Press ENTER to exit console.
```

Practical 24

Aim:
To find the sum of two matrices.

Description:
Matrix addition means to add two matrices, i.e., compute their sum and print it. A user inputs the elements and the matrices are formed. Then we input the formula for the addition, and print the sum of the matrix.

Program:

```
#include <stdio.h>

int main()
{
  int i, j, a[10][10], b[10][10], sum[10][10];


  printf("Enter the elements of first matrix\n");
  for(i=0;i<2;i++)
  {
    for(j=0;j<2;j++)
    {
      scanf("%d", &a[i][j]);
    }
  }
  printf("Enter the elements of second matrix\n");
  for(i=0;i<2;i++)
  {
    for(j=0;j<2;j++)
    {
      scanf("%d", &b[i][j]);
    }
```

```
    }
    printf("Sum of entered matrices:-\n");
    for(i=0;i<2;i++)
    {
      for(j=0;j<2;j++)
      {
        sum[i][j] =a[i][j]+b[i][j];
        printf("%d\t", sum[i][j]);
      }
      printf("\n");
    }
    return 0;
}
```

Output:

```
Enter the elements of first matrix
1
2
1
2
Enter the elements of second matrix
1
2
1
2
Sum of entered matrices:-
2       4
2       4


...Program finished with exit code 0
Press ENTER to exit console.
```

Practical 25

Aim:
To find the difference of two matrices.

Description:
Matrix subtraction means to subtract two matrices, i.e., compute their difference and print it. A user inputs the elements and the matrices are formed. Then we input the formula for the subtraction, and print the difference of the matrix.

Program:

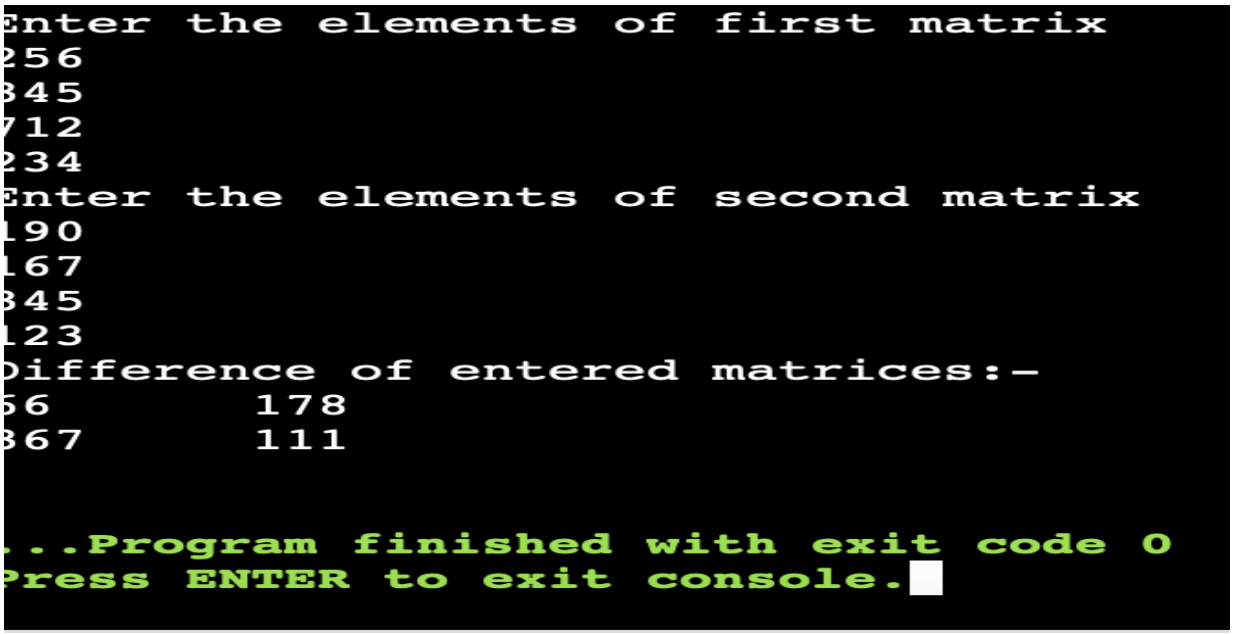
```
#include <stdio.h>

int main()
{
  int i, j, a[10][10], b[10][10], diff[10][10];


  printf("Enter the elements of first matrix\n");
  for(i=0;i<2;i++)
  {
    for(j=0;j<2;j++)
    {
      scanf("%d", &a[i][j]);
    }
  }
  printf("Enter the elements of second matrix\n");
  for(i=0;i<2;i++)
  {
    for(j=0;j<2;j++)
    {
      scanf("%d", &b[i][j]);
```

```
        }
    }
    printf("Difference of entered matrices:-\n");
    for(i=0;i<2;i++)
    {
        for(j=0;j<2;j++)
        {
            diff[i][j] =a[i][j]-b[i][j];
            printf("%d\t", diff[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Output:



Practical 26

Aim:
To multiply two matrices.

Description:
The user inputs the elements of the two matrices and then we are performing multiplication on the matrices entered by the user. In matrix multiplication, the first matrix of one row element is multiplied by the second matrix of all column elements.

Program:

```
#include <stdio.h>

int main()
{
  int i, j, a[10][10], b[10][10], product=0, c[10][10],k;


  printf("Enter the elements of first matrix\n");
  for(i=0;i<2;i++)
  {
    for(j=0;j<2;j++)
    {
      scanf("%d", &a[i][j]);
    }
  }
  printf("Enter the elements of second matrix\n");
  for(i=0;i<2;i++)
  {
    for(j=0;j<2;j++)
    {
      scanf("%d", &b[i][j]);
    }
```

```
    }
    printf("product of entered matrices:-\n");
    for(i=0;i<2;i++)
    {
       for(j=0;j<2;j++)
       {
         for(k=0;k<2;k++)
         {
          product=product+a[i][k]*b[k][j];
         }
         c[i][j]=product;
         product=0;
       }
    }
    for(i=0;i<2;i++)
    {
        for(j=0;j<2;j++)
        {
            printf("%d\t", c[i][j]);
        }
        printf("\n");

    }
    return 0;
}
```

Output:

```
Enter the elements of first matrix
1
1
2
2
Enter the elements of second matrix
1
2
1
2
product of entered matrices:-
2       4
4       8

...Program finished with exit code 0
Press ENTER to exit console.
```

Practical 27

Aim:
To read a series of words using scanf().

Description:
We will create four strings- word1, word2, word3 and word4. By using 'scanf' we can let users input the characters. And the printing of words is done by 'printf'

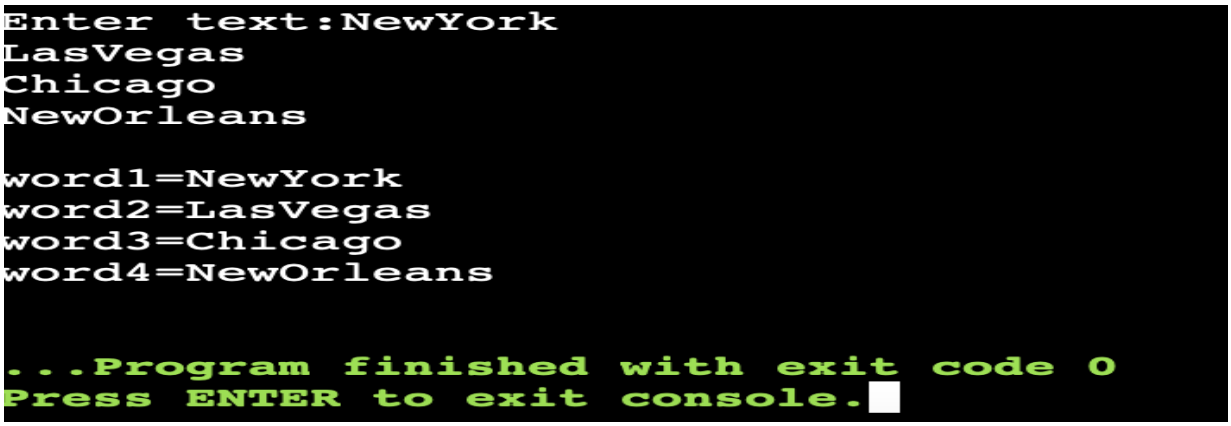
Program:

```
#include<stdio.h>

int main()
{
   char word1[40], word2[40], word3[40], word4[40];

   printf("Enter text:");
   scanf("%s%s",word1, word2);
   scanf("%s",word3);
   scanf("%s",word4);
   printf("\n");
   printf("word1=%s\nword2=%s\n",word1,word2);
   printf("word3=%s\nword4=%s\n",word3,word4);
   return 0;
}
```

Output:


Enter text:NewYork
LasVegas
Chicago
NewOrleans
word1=NewYork
word2=LasVegas
word3=Chicago
word4=NewOrleans
...Program finished with exit code 0
Press ENTER to exit console.

Practical 28

Aim:

To copy one string into another with and without using function.

I. Using strcpy() function:

Description:

The strcpy() function is a built-in library function, declared in the string.h header file. It copies the source string to the destination string until a null character (\0) is reached.

Program:

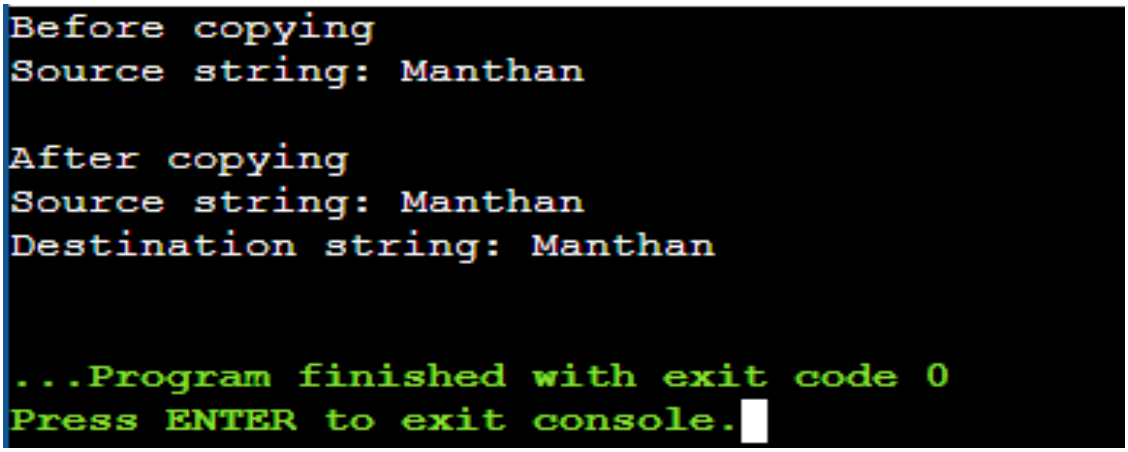
```
#include <stdio.h>
#include <string.h>

int main()
{
  char a[15]=
  "Manthan"; char
  b[15];
  printf("Before copying\n");
  printf("Source string: %s \n", a);

  strcpy(b,a);
  printf("\nAfter copying\n");
  printf("Source string: %s \n", a);
  printf("Destination string: %s \n", b);

  return 0;
}
```

Output:



II. Without using strcpy() function:

Description:

We are giving one string in input and then with the help of a 'for' loop we transfer the content of the first array to the second array.

Program:

```
#include<stdio.h>
#include<string.h>
int main()
{
   char
   a[15]="Manthan";
   char b[15], i;
   printf("The original string is: %s\n", a);
   for(i=0;a[i]!='\0';i++)
   {
      b[i]=a[i];
   }
   b[i]='\0';
   printf("The copied string is: %s\n", b );
   return 0; }
```

Output:

```
The original string is: Manthan
The copied string is: Manthan


...Program finished with exit code 0
Press ENTER to exit console.
```

Practical 29

Aim:
To concatenate strings with and without functions.

I. With strcat() function.

Description:
The concatenation of strings is a process of combining two strings to form a single string. If there are two strings, then the second string is added at the end of the first string.

Program:

```
#include<stdio.h>
#include<string.h>
int main()
{
   char a[20], b[20];
   printf("Enter string a: \n");
   scanf("%s", a);
   printf("Enter string b: \n");
   scanf("%s", b); strcat(a,b);
   printf("The concatenated string is: \n%s", a);
   return 0;
}
```

Output:

```
Enter string a:
Manthan
Enter string b:
Chandak
The concatenated string is:
ManthanChandak

...Program finished with exit code 0
Press ENTER to exit console.
```

II. Without strcat() function.

Description:

Get the two Strings to be concatenated. Declare a new Strings to store the concatenated String. Insert the first string in the new string. Insert the second string in the new string. Print the concatenated string

Program:

```
#include<stdio.h>

int main()
{
   char a[20] = "Manthan", b[100] =
   "Chandak"; char c[100];
   int i=0, j=0;
   printf("\nFirst string: %s", a);
   printf("\nSecond string: %s",
   b); while (a[i] != '\0')
   {
      c[j] =
      a[i]; i++;
      j++;
```

```
    }
    i = 0;
    while (b[i] != '\0')
    {
        c[j] =
        b[i]; i++;
        j++;
    }
    c[j] = '\0';
    printf("\nConcatenated string: %s",
    c); return 0;
}
```

Output:

```
First string: Manthan
Second string: Chandak
Concatenated string: ManthanChandak

...Program finished with exit code 0
Press ENTER to exit console.
```

Practical 30

Aim:
To swap two numbers using functions.

Description:
Swapping two variables refers to mutually exchanging the values of the variables. Generally, this is done with the data in memory. We will use the method of swapping two variables using a third temporary variable.

Program:

```
#include<stdio.h>
void swap(int,int);
int main()
{
   int
   a,b;
   a=10;
   b=77;
   printf("The value before swapping is: %d and %d",a,b);
   printf("\n");
   swap(a,b);
   printf("\n");
   printf("The value after swapping is: %d and %d", a,b);
}
void swap(int x, int y)
{
   int
   t;
   t=x;
   x=y;
   y=t;
   printf("The swap of a and b is %d and %d", x,y);
}
```

Output:

```
The value before swapping is: 10 and 77
The swap of a and b is 77 and 10
The value after swapping is: 10 and 77

...Program finished with exit code 0
Press ENTER to exit console.
```

Practical 31

Aim:
To find the factorial of a given number using functions.

Description:
Take input 'a' for which you want to calculate it's factorial for. Let us take n=4. 1st iteration for(i=1;i<=n;i++) i.e for(i=1;i<=4;i++) as n=4. fact=fact*i i.e fact=1*1 as fact is initalized to 1 and i=1 hence fact=1. 2nd iteration for(i=2;i<=n;i++) i.e for(i=2;i<=4;i++). {fact=fact*i} i.e., fact=1*2 hence fact=2. 3rd iteration for(i=3;i<=n;i++) i.e for(i=3;i<=4;i++). fact=fact*i i.e fact=2*3 hence fact=6. 4th iteration for(i=4;i<=n;i++) i.e for(i=4;i<=4;i++). fact=fact*i i.e., fact=6*4 hence fact=24. 'For' loop ends here. We print the output.

Program:

```
#include <stdio.h>
int fact(int);
void main()
{
 int a,factorial;

        printf("Enter a number to calculate it's factorial\n");
        scanf("%d",&a);
        factorial=fact(a);
    printf("Factorial of the num(%d) = %d\n",a,factorial);
}
int fact(int x)
{
    int i,f=1;
```

```
    for(i=1;i<=x;i++)
    {
        f=f*i;
    }
    return f;
}
```

Output:

```
Enter a number to calculate it's factorial
4
Factorial of the num(4) = 24


...Program finished with exit code 0
Press ENTER to exit console.
```

Practical 32

Aim:
To show the table of a given number using functions.

Description:
Here, the user input is stored in the int variable n. Then, we use a for loop to print the multiplication table up to 10. The loop runs from i = 1 to i = 10. In each iteration of the loop, n * i is printed.

Program:

```
#include<stdio.h>
void main()
{
	void table();
	table();
}
void table()
{
	int x,i,a;
	printf("Enter a number to know table: ");
	scanf("%d",&x);

	for(i=1;i<=10;i++)
	{
		a=x*i;

		printf("%d*%d=%d\n",x,i,a);
	}
}
```

Output:

```
Enter a number to know table: 55
55*1=55
55*2=110
55*3=165
55*4=220
55*5=275
55*6=330
55*7=385
55*8=440
55*9=495
55*10=550


...Program finished with exit code 0
Press ENTER to exit console.
```

Practical 33

Aim:
To find whether the given number is odd or even.

Description:
The integer entered by the user is stored in the variable 'x'. Then, whether x is perfectly divisible by 2 or not is checked using the modulus % operator. If the x is perfectly divisible by 2, test expression x%2 == 0 evaluates to 1 (true). This means x is even. However, if the test expression evaluates to 0 (false), the x is odd.

Program:

```
#include<stdio.h>
int OddEven(intx);
int main()
{
    int x;
    printf("Enter a number: ");
    scanf("%d",&x); OddEven(x);
    return 0;
}
int OddEven(int a)
{
    if(a%2==0)
        printf("%d is even.", a);
    else
        printf("%d is odd.", a);
}
```

Output:

```
Enter a number: 66
66 is even.

...Program finished with exit code 0
Press ENTER to exit console.
```

Practical 34

Aim:
To find whether the given number is a prime number or not.

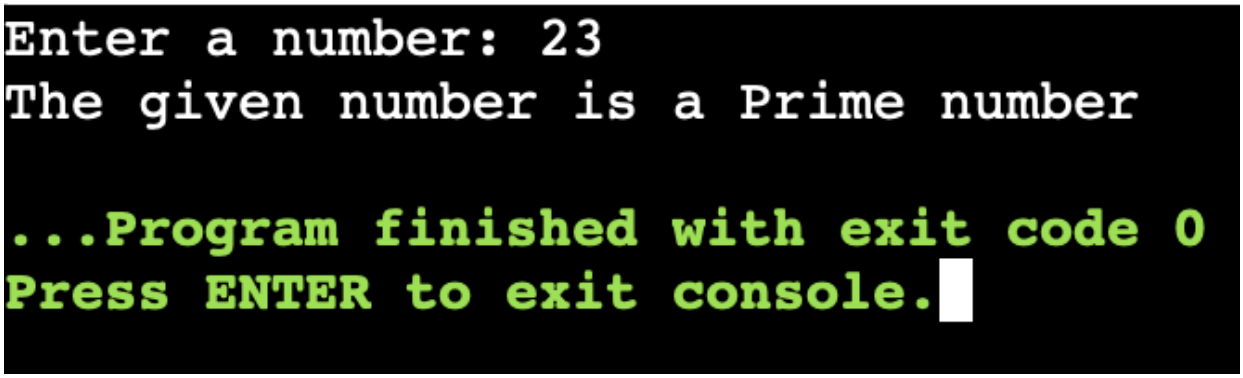
Description:
Here we will check whether the given is a prime number or not. If the number entered is prime, then we will print 'The given number is prime' and if the number isnt prime, we will print 'The given number isn't prime'.

Program:

```
#include <stdio.h>
int Prime(int x);
int main()
{
    int x;
    printf("Enter a number: ");
    scanf("%d",&x);
    Prime(x);
    return 0;
}
int Prime(int a)
{
    int x, i, c = 0;
    for(i=1;i<=a;i++)
    {
      if(a%i==0)
      {
         c++;
      }
    }
    if(c==2)
```

```
    {
    printf("The given number is a Prime number");
    }
    else
    {
    printf("The given number is not a Prime number");
    }
}
```

Output:



Practical 35

Aim:
To swap numbers using call by value.

Description:
Here we will save the value in a third variable and then equate it with the second number.

Program:

```
#include <stdio.h>
void swap(int,int);
void main()
{
   int x,y;
   printf("Enter two numbers: ");
   scanf("%d%d",&x,&y);
   printf("The numbers before swapping are: %d and %d\n", x,y); swap(x,y);
}
void swap(int a, int b)
{
   int
   t;
   t=a;
   a=b;
   b=t;
   printf("The numbers after swapping are: %d and %d\n", a,b);

}
```

Output:

```
Enter two numbers: 44
77
The numbers before swapping are: 44 and 77
The numbers after swapping are: 77 and 44


...Program finished with exit code 0
Press ENTER to exit console.
```

Practical 36

Aim:
To swap numbers using call by reference.

Description:
Same as the last practical, but here we will use the pointers for the reference of the numbers.

Program:

```
#include <stdio.h>
void swap(int*, int*);
void main()
{
    int x,y;
    printf("Enter two numbers: ");
    scanf("%d%d",&x,&y);
    printf("The numbers before swapping are: %d and %d\n", x,y); swap(&x,&y);
}
void swap(int*a, int*b)
{
    int t;
    t=*a;
    *a=*b;
    *b=t;
    printf("The numbers after swapping are: %d and %d\n", *a,*b);

}
```

Output:

```
Enter two numbers: 33
55
The numbers before swapping are: 33 and 55
The numbers after swapping are: 55 and 33


...Program finished with exit code 0
Press ENTER to exit console.
```

Practical 37

Aim:
To find the largest among two numbers using functions.

Description:
We ask the user to enter 2 integer numbers. We pass those 2 integer numbers to the user defined function 'Large'. Inside the function 'large' we use 'if and else' to determine the biggest number. Function biggest returns the biggest of the 2 numbers. Here if x is greater than y, x will be returned else y will be returned.

Program:

```
#include <stdio.h> int
findlargest(int,int); int
main()
{
   int a,b,large;
   printf("Enter two numbers:\n");
   scanf("%d%d", &a,&b);
   printf("\nThe two numbers are: %d and %d\n", a,b);
   large=findlargest(a,b);
   printf("\nLargest is %d\n", large);
}
int findlargest(int x, int y)
{
   if(x>y)
   {
      return x;
   }
   else
   {
      return y;
```

```
    }
}
```

Output:

```
Enter two numbers:
33
-99

The two numbers are: 33 and -99

Largest is 33


...Program finished with exit code 0
Press ENTER to exit console.
```

Practical 38

Aim:

To pass an array from main function to display function (one-dimensional).

Description:

To pass an entire array to a function, only the name of the array is passed as an argument. [] is to inform the compiler that you are passing a one-dimensional array to the function.

Program:

```
#include <stdio.h>
void print(int a[10]);
 int main()
{
   int i,a[10];
   printf("Enter array elements:
   \n"); for(i=0;i<10;i++)
   {
      scanf("%d", &a[i]);
   }
   print(a);
}
void print(int a[10])
{
   int i;
   printf("The elements
   are:\n"); for(i=0;i<10;i++)
   {
      printf("%d\t", a[i]);
   }
   printf("\n");
}
```

Output:

```
Enter array elements:
1
2
1
2
1
2
1
2
1
2
The elements are:
1       2       1       2       1       2       1       2       1       2


...Program finished with exit code 0
Press ENTER to exit console.
```

Practical 39

Aim:

To pass an array from main function to display function (Two-dimensional).

Description:

Same as the previous practical but in Two-Dimension and using two 'for' loops for the two dimensions.

Program:

```
#include  <stdio.h>
void print(int a[10][10]);
int main()
{
    int i,j,a[10][10];
    printf("Enter array elements: \n");
    for(i=0;i<2;i++)
    {
        for(j=0;j<2;j++)
        {
        scanf("%d", &a[i][j]);
        }
    }
    print(a);
}
void print(int a[10][10])
{
    int i,j;
    printf("The elements are:\n");
    for(i=0;i<2;i++)
    {
        for(j=0;j<2;j++)
```

```
        {
        printf("%d\t", a[i][j]);
        }
        printf("\n");
    }
}
```

Output:

```
Enter array elements:
1
2
1
2
The elements are:
1           2
1           2


...Program finished with exit code 0
Press ENTER to exit console.
```

Practical 40

Aim:
To display the sum of array elements using arrays with functions.

Description:
Declare a variable to store the sum. Say int sum;. We should initialize the sum variable to 0 .i.e. sum=0; . Loop through all the elements in the array and add them to variable sum. Print the sum.

Program:

```
#include<stdio.h>
int Sum(int a[10], int n);
int main()
{
 int i, n, a[10];
 int sum;
 printf("Please Enter the Size of an Array: "); scanf("%d",
 &n);
 printf("\nPlease Enter Array Elements\n");
 for(i=0;i<n;i++)
  {
      scanf("%d", &a[i]);
  }
  sum=Sum(a, n);
 printf("Sum of All Elements in an Array = %d ", sum);
 return 0;
}
int Sum(int a[10], int n)
{
        int sum=0;
        int i;
        for(i=0;i<n; i++)
```

```
        {
      sum=sum+a[i];
        }
         return sum;
}
```

Output:

```
Please Enter the Size of an Array: 3

Please Enter Array Elements
1
2
3
Sum of All Elements in an Array = 6

...Program finished with exit code 0
Press ENTER to exit console.
```

Practical 41

Aim:
To show the basic declaration of pointers.

Description:
A pointer is a special type of variable which stores the address of a variable.

Program:

```
#include<stdio.h>
int main()
{
   int a=10;
   int *p; //pointer declaration. p=&a;
   //pointer initialization.
//here p is holding address pf the variable a.
   printf("%d\n", a);
   printf("%d\n", *p);
   return 0;
}
```

Output:

```
10
10


...Program finished with exit code 0
Press ENTER to exit console.
```

Practical 42

Aim:
To add two numbers usingpointers.

Description:
We have two integer variables x and y and two pointer variables p and q. We assign the addresses of x and y to p and q respectively and then assign the sum of x and y to the variable sum. Remember '&' is the address of operator and '*' is valued at the address operator.

Program:

```
#include <stdio.h>
int main()
{
   int first, second, *p, *q, sum;
   printf("Enter two integers to add\n");
   scanf("%d%d", &first, &second); p=&first;
   q=&second;
   sum=*p+*q;
   printf("Sum of the numbers=%d\n", sum);
   return 0;
}
```

Output:

```
Enter two integers to add
34
66
Sum of the numbers = 100


...Program finished with exit code 0
Press ENTER to exit console.
```

Practical 43

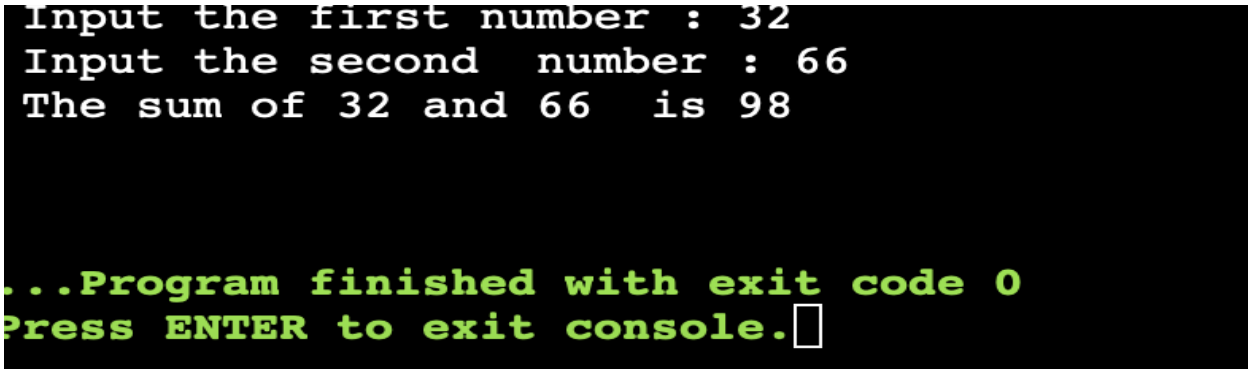
Aim:
To add two numbers by using call by reference.

Description:
The program accepts 2 numbers from the user and stores the value in 'a' and 'b'. The program then makes a call to 'add function'. We are passing address/reference of 'a' and 'b' as function parameters and hence the name, call by reference (also known as – pass by reference).

Program:

```
#include <stdio.h>
int add(int*, int*);
int main()
{
   int a, b, sum;
   printf(" Input
   th
e first number : ");
   scanf("%d", &a);
   printf(" Input the second number : ");
   scanf("%d", &b);
   sum=add(&a, &b);
   printf(" The sum of %d and %d is %d\n\n", a, b, sum);
   return 0;
}
int add(int *n1, int *n2)
{
   int sum;
   sum=*n1+*n2;
   return sum;
}
```

Output:


Input the first number : 32
Input the second number : 66
The sum of 32 and 66 is 98
...Program finished with exit code 0
Press ENTER to exit console.

Practical 44

Aim:

To store 'n' elements in an array and print the elements using pointer.

Description:

In this the elements of the array are stored in a[25]. The elements are accessed by the '*' notation.

I. a[0] is equivalent to *a and &a[0].
II. a[1] is equivalent to *(a+1) and &a[1].
III. a[2] is equivalent to *(a+2) and &a[2].

….. And so on.

Program:

```
#include <stdio.h>
int main()
{
  int a[25], i,n;
  printf("Input the number of elements to store in the array :"); scanf("%d",&n);
  printf("Input %d elements in the array :\n",n);
  for(i=0;i<n;i++)
  {
     printf("element-[%d]: ",i);
     scanf("%d",a+i);
  }
  printf("The elements you entered are:\n");
  for(i=0;i<n;i++)
   {
      printf("element-[%d]:[%d]\n",i,*(a+i));
   }
```

```
        return 0;
}
```

Output:

```
Input the number of elements to store in the array :5
Input 5 elements in the array :
element-[0]: 1
element-[1]: 2
element-[2]: 3
element-[3]: 4
element-[4]: 5
The elements you entered are:
element-[0]:[1]
element-[1]:[2]
element-[2]:[3]
element-[3]:[4]
element-[4]:[5]


...Program finished with exit code 0
Press ENTER to exit console.
```

Practical 45

Aim:
To compute the sum of all elements in an array using pointers.

Description:
The sequence of steps for the solution will be as follows:

1. Create a pointer variable, which points to an integer data.
2. Take a number i.e size of array as input.
3. Create a block of space fo size (size-of-array*sizeof(int)) assigning the starting address of this whole space to the pointer variable.
4. Iterate via forloop for reading array elements as input.
5. Iterate again via forloop to access the address stored in pointer variable, adding the iterator value to it, so as to access every element of array, and calculating the overall sum.

Program:

```
#include <stdio.h>
void main()
{
   int a[10];
   int i,n, sum = 0;
   int *p;
   printf("Input the number of elements to store in the array: "); scanf("%d",&n);
   printf("Input %d number of elements in the array:\n",n);
   for(i=0;i<n;i++)
    {
          printf("element - %d : ",i+1);
          scanf("%d",&a[i]);
    }
```

```
    p=a;
    for (i = 0; i < n; i++)
    {
     sum=sum+*p;
     p++;
    }
    printf("The sum of array is: %d\n\n", sum);
}
```

Output:

```
Input the number of elements to store in the array: 3
Input 3 number of elements in the array:
element - 1 : 456
element - 2 : 445
element - 3 : 447
The sum of array is: 1348


...Program finished with exit code 0
Press ENTER to exit console.
```