**Recurssion**

1. Write at least 3 paragraphs about recursion and its process.

A function that calls itself is known as recursive function. And, this technique is known as recursion. Recursion is the process of repeating items in a self-similar way. In programming languages, if a program allows us to call a function inside the same function, then it is called a recursive call of the function. The C programming language supports recursion, i.e., a function to call itself. But while using recursion, programmers need to be careful to define an exit condition from the function, otherwise it will go into an infinite loop.

Recursive functions are very useful to solve many mathematical problems, such as calculating the factorial of a number, generating Fibonacci series, etc.

Recursion is a programming technique that allows the programmer to express operations in terms of themselves. In C, this takes the form of a function that calls itself. A useful way to think of recursive functions is to imagine them as a process being performed where one of the instructions is to "repeat the process".

1. WAP to find multiplication of N numbers

#include<stdio.h>

int mult(int);

int main()

{

int n,m;

printf("Enter the number:");

scanf("%d",&n);

m=mult(n);

printf("The multiplication is: %d",m);

return 0;

}

int mult(int n)

{

int m;

if(n==0){

return 1;

}

else{

m=n\*mult(n-1);

return f;

}

}



1. WAP to find factorial of N number

#include<stdio.h>

int fact(int);

int main()

{

int n,f;

printf("Enter the number:");

scanf("%d",&n);

f=fact(n);

printf("The factorial is: %d",f);

return 0;

}

int fact(int n)

{

int f;

if(n==0){

return 1;

}

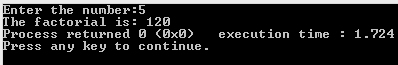
else{

f=n\*fact(n-1);

return f;

}

}



1. WAP to find Fibonacci series in c using for loop

#include<stdio.h>

void main()

{

int n,a=0,b=1,c=0,i;

printf("Enter the number to which you want to find a series:");

scanf("%d",&n);

printf("Fibonacci series:%d,%d,",a,b);

for(i=3;i<=n;++i)

{

c=a+b;

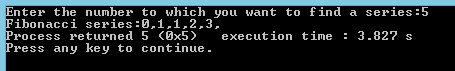
a=b;

b=c;

printf("%d,",c);

}

}



**Pointer**

1. Differentiate between Passing by Value and Passing by Reference, write programs for each.

Passing by value:

Pass by value: When values of actual arguments are passed to a function as arguments, it is known as function call by value. In this type of call, the value of each actual arguments is copied into corresponding formal argument of the function definition. The content of the arguments in the calling function are not altered, even if they are changed in the called function. The function is called using the syntax:

function\_name(value\_of\_argument1,value\_of\_argument2,…);

Example:

#include<stdio.h>

void swap(int, int);

void main()

{

int a,b;

a=99,b=89;

printf("\nBefore function calling, a and b are:%d\t%d",a,b);

swap(a,b);

printf("\n After function calling, a and b are: %d\t%d",a,b);

getch();

}

void swap(int x, int y)

{

int temp;

temp=x;

x=y;

y=temp;

printf("\n THe values within functions are :%d\t%d",x,y);

}

Pass by reference- In this type of function call, the address of a variable or argument is passed to a function as argument instead of actual value of variable. A function can be called by passing address in its argument as:

function\_name(address\_of\_argument1, address\_of\_argument2,…);

Example:

#include<stdio.h>

void swap(int \*, int \*);

void main()

{

int a,b;

a=99,b=89;

printf("\nBefore function calling, a and b are:%d\t%d",a,b);

swap(&a,&b);

printf("\n After function calling, a and b are: %d\t%d",a,b);

getch();

}

void swap(int \*x, int \*y)

{

int temp;

temp=\*x;

\*x=\*y;

\*y=temp;

}

1. WAP to print the sum of two matrix using dynamic memory Allocation (STRICTLY USE DMA)

#include<stdio.h>

void main()

{

int i,j,p,q,r,s,\*m1, \*m2, \*a;

printf("Enter the order of matrix A:\n");

scanf("%d%d", &p,&q);

printf("\nEnter the order of matrix B:\n");

scanf("%d%d", &r,&s);

if(q==r)

{

m1=(int\*)calloc(p\*q, sizeof(int));

m2=(int\*)calloc(r\*s, sizeof(int));

a=(int\*)calloc(p\*s, sizeof(int));

printf("\nEnter the elements of matrix A:\n");

for(i=0;i<p;i++)

{

for(j=0;j<q;j++)

{

scanf("%d",(m1+i\*q+j));

}

}

printf("\nEnter the elements of matrix B:\n");

for(i=0;i<r;i++)

{

for(j=0;j<s;j++)

{

scanf("%d",(m2+i\*r+j));

}

}

printf("\nAddition:");

for(i=0;i<p;i++)

{

for(j=0;j<q;j++)

{

\*(a+i\*q+j)=\*(m1+i\*q+j)+(\*(m2+i\*r+j));

}

}

printf("\nSum of the Matrices A and B:");

for(i=0;i<p;i++)

{

printf("\n");

for(j=0;j<q;j++)

{

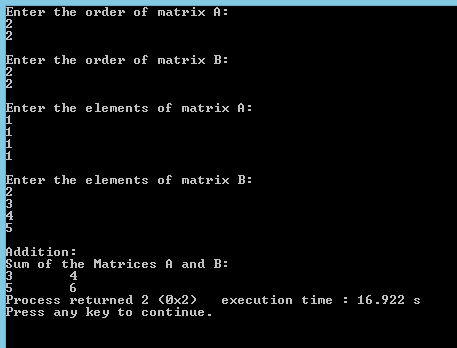
printf("%d\t",\*(a+i\*q+j));

}

}

}

}



**Algorithm**

1. Why an algorithm is necessary before trying to solve any problem?

An algorithm is step by step description of the method to solve a problem. It is an effective procedure for solving a problem in a finite number of steps. Algorithm maintains sequences of computer instructions required to solve a problem in such a way that if the instructions are executed in the specified sequence, the desired result is obtained. Developing an algorithm is a step of program design hence it is a must.

1. Explain one of your real life problem and design an algorithm to solve it.

Algorithm for making a tea:

Step1: Start

Step2: Get up from bed.

Step3: Fill the kettle with water.

Step4: Turn the gas stove on.

Step5: Put the kettle on the stove.

Step6: Put sugar and tea in the kettle.

Step7: If the tea in the kettle is not boiling, then goto step7

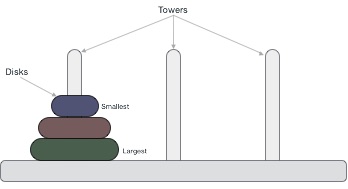
Step8: Switch the gas off.

Step9: Pour tea from the kettle into cup.

Step10: Stop

**MISSING GAP**

1. Research about Tower of Hanoi using text books or internet. Write what you can understand; also suggest how can we solve a tower of hanoi problem (ALSO write an algorithm)

Tower of Hanoi, is a mathematical puzzle which consists of three tower (pegs) and more than one rings

These rings are of different sizes and stacked upon in ascending order i.e. the smaller one sits over the larger one. There are other variations of puzzle where the number of disks increase, but the tower count remains the same.

Rules

The mission is to move all the disks to some another tower without violating the sequence of arrangement. The below mentioned are few rules which are to be followed for tower of hanoi −

* Only one disk can be moved among the towers at any given time.
* Only the "top" disk can be removed.
* No large disk can sit over a small disk.

Algorithm

Step 1 − Move n-1 disks from source to aux

Step 2 − Move nth disk from source to dest

Step 3 − Move n-1 disks from aux to dest