RECURSIVE FUNCTIONS

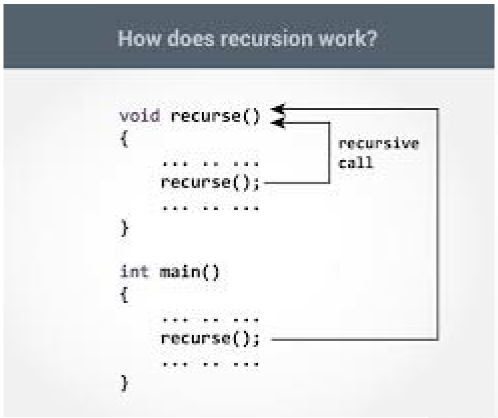
Recursion is the process of repeating items in a self-similar way. In programming languages, if a program allows you to call a function inside the same function, then it is called a recursive call of the function.Recursion is used to solve various mathematical problems by dividing it into smaller problems. In programming , it is used to divide complex problem into simpler ones and solving them individually.But while using recursion, we need to be careful to define an exit condition from the function, otherwise it will go into an infinite loop.Types of recursion:

1) Direct recursion

2) Indirect recursion

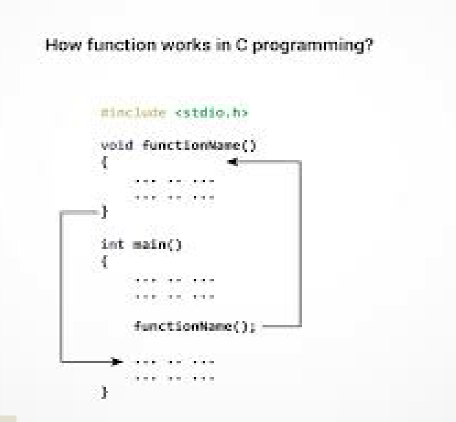
Direct Recursion:A function is said to be direct recursive if it calls itself directly.

Indirect Recursion:A function is said to be indirect recursive if it calls another function and this new function calls first calling function again.



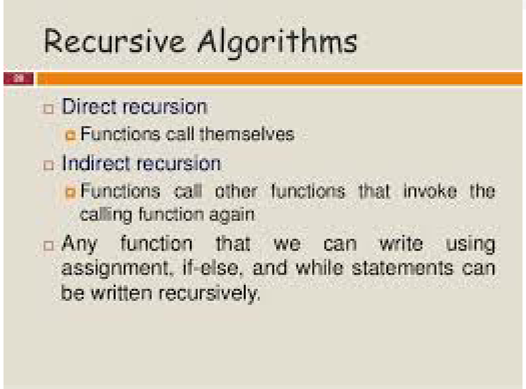
How does recursion work?

```
void recurse()
{
    ... .. ...
    recurse();
    ... .. ...
}

int main()
{
    ... .. ...
    recurse();
    ... .. ...
}
```



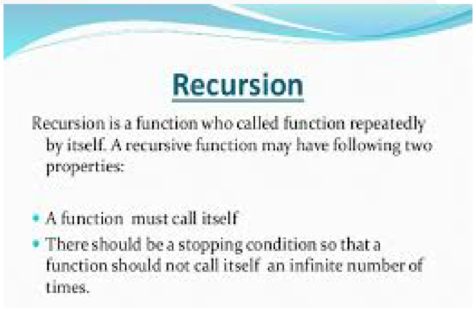
How function works in C programming?

```
#include <stdio.h>

void functionName()
{
    ... .. ...
    ... .. ...
}

int main()
{
    ... .. ...
    ... .. ...

    functionName();

    ... .. ...
    ... .. ...
}
```



Recursive Algorithms

- Direct recursion
  - Functions call themselves
- Indirect recursion
  - Functions call other functions that invoke the calling function again
- Any function that we can write using assignment, if-else, and while statements can be written recursively.

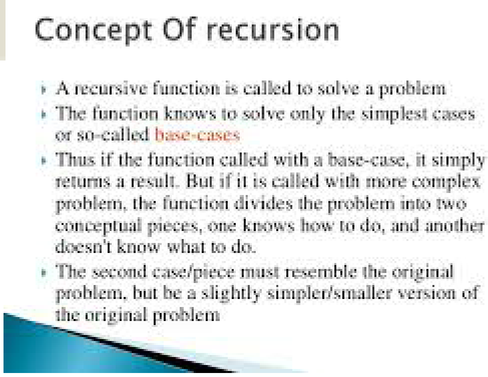


Recursion

Recursion is a function who called function repeatedly by itself. A recursive function may have following two properties:

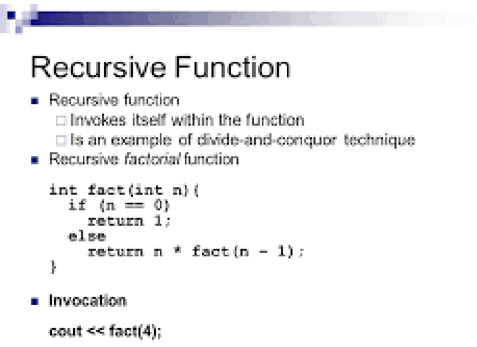
- A function must call itself
- There should be a stopping condition so that a function should not call itself an infinite number of times.

Advantages and Disadvantages of Recursion:Recursion makes program elegant and cleaner. All algorithms can be defined recursively which makes it easier to visualize and prove.If the speed of the program is vital then, you should avoid using recursion. Recursions use more memory and are generally slow. Instead, you can use loop.Factorial program:



Concept Of recursion

- A recursive function is called to solve a problem
- The function knows to solve only the simplest cases or so-called base-cases
- Thus if the function called with a base-case, it simply returns a result. But if it is called with more complex problem, the function divides the problem into two conceptual pieces, one knows how to do, and another doesn't know what to do.
- The second case/piece must resemble the original problem, but be a slightly simpler/smaller version of the original problem



Recursive Function

- Recursive function
  - □ Invokes itself within the function
  - □ Is an example of divide-and-conquor technique
- Recursive *factorial* function

```
int fact(int n){
   if (n == 0)
      return 1;
   else
      return n * fact(n - 1);
}
```

- Invocation

```
cout << fact(4);
```

```
int Factorial(int n) {

// Simple case: 0! = 1

if (n == 0) return 1;// General function: n! = n * (n -1)!

return (n * Factorial(n - 1));}
```

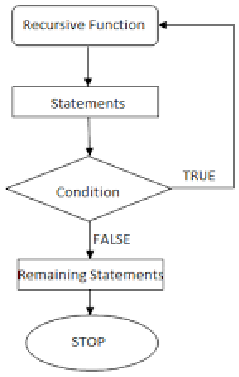


Fig: Flowchart showing recursion