Recursion:

- Recursion in computer science is a problem solving method where the solution to a problem depends on solutions to smaller instances of the same problem.
- This approach can be applied to many types of problems and is one of the central ideas of computer science.
- Most computer programming languages support recursion by allowing a function to call itself within the program text.

Problem solving with Recursion:

- To solve a problem using recursive method two conditions must be satisfied:

1) Problem should be written or defined in ~~terms~~ terms of its previous result.
2) problem statement must include a terminating condition otherwise the function will never terminate. This means that there must be an if statement somewhere in the recursive function to force the function to return without the recursive call being executed.

~~Recursion~~ Recursion versus Iteration:

Recursion	Iteration
1. A function is called from the definition of the same function to do repeated task.	1. Loops are used to perform repeated task.
2. Recursion is a top-down approach to problem solving; it divides the problem into pieces.	2. It is a bottom-up approach: it begins from what is known & from this it construct the sol^{n} step-by-step.

Eg; Computing factorial of num.
`long int factorial (int n)`
`{`
`if (n==0) return 1;`
`else`
`return (n*factorial (n-1));`
`}`
	Eg; Computing factorial of a num.
	`int fact=1;`
	`for(i=1; i<=n; i++)`
	`{`
	`fact= fact*i;`
	`}`

3. Problem to be solved is defined in term of its previous result to solve a problem using recursion.	3. It is not necessary to define a problem in term of its previous result to solve using iteration. For eg: "display ur name 1000 times"
4. In recursion, a function calls to itself until some conditions is satisfied.	4. In iteration, a function doesnot call to itself.
5. All problems cannot be solved using recursion.	5.) All problems can be solved using iteration,
6. Recursion utilizes stack.	6) Iteration docsnot utilizes stack.

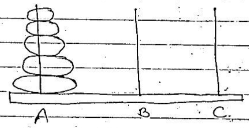
Uses of Stack in Recursion:

- Recursion uses stack to keep the successive generations of local variables and parameters of the function in its corresponding calls.
- This stack is maintained by the C system and is invisible to the user (programmer).
- Each time a recursive function is entered, a new allocation of its variables is pushed on top of the stack.
- Any reference to a local variable or parameter is through the current top of the stack.
- When the function returns, the stack is popped, the top allocation is freed and the previous allocation becomes the current stack top to be used ~~to be~~ for referencing local variables.
- Each time a recursive function returns, it returns to the point immediately following the point from which it was called.

Q. What do you mean by recursion? Explain the implementation of factorial and fibbonacci sequences with example.

"Towers of Hanoi" problem:

- There are 3 pegs A, B, C.
- Five disks (say) of different diameters are placed on peg A so that a larger disk is always below a smaller disk.
- The aim is to move the five disks to peg C, using peg B as auxiliary.
- Only the top disk on any peg may be moved to any other peg, and a larger disk may never rest on a smaller one.



Q. Algorithm to solve the problem of "tower of hanoi" to move n disk from peg A to peg C using B as auxilliary:

1. If n=1 move the single disk from A to C and stop.
2. Move top n-1 disks from A to B using C as auxiliary.
3. Move the remaining disk from A to C.
4. Move the n-1 disk from B to C using A as auxiliar

A-C, A-B, C-B, A-C, B-A, B-C, [illegible]
A to C.