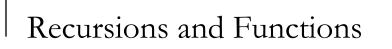


- Next class on next Wednesday as usual
- First class test on Wednesday (6th Sep.)
 - □ 1 hour duration (6pm-7pm)
 - Please assemble half an hour early



What are recursions?

- Expressing an entity in terms of itself is called recursion.
- But remember it is not a circular definition.
 - here a solution to a bigger problem is expressed in terms of solutions to smaller problems.
 - a general mathematical concept



- The factorial of 0 is 1, and the factorial of any positive integer n is the product of all integers from 1 to n.
- The factorial of 0 is 1. The factorial of any positive integer n is the product of n and the factorial of (n-1).
 - the first definition is iterative, while the second one is recursive.
 - The factorial of (n-1) is solved in the same way, until the factorial of 0 is asked for, which is 1.

```
The C function

int factorial ( int n )

{

    if (n < 0) return (-1); /* Error condition */

    if (n == 0) return (1); /* Base case */

    return(n * factorial(n-1)); /* Recursive call */

}
```

A recursive function

- A function that calls itself, or which calls another function, which calls the first one is called recursive function.
- The later definition can be extended to more than two functions.
- However there are two important criteria:
 - Each time a function call occurs, it must be closer, in some sense to the solution.
 - There must be a decision criteria for stopping the process or computation: this being called "the escape hatch".

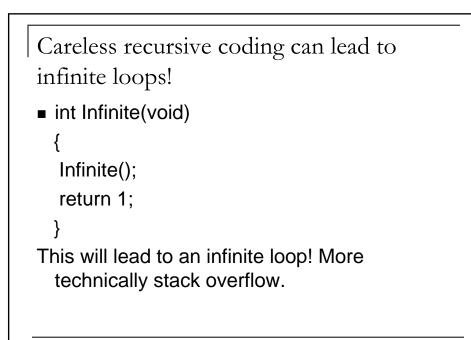
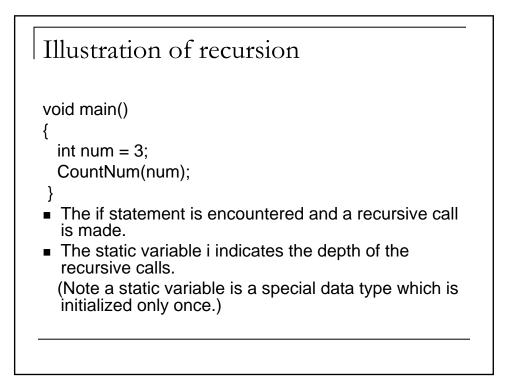
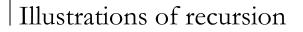


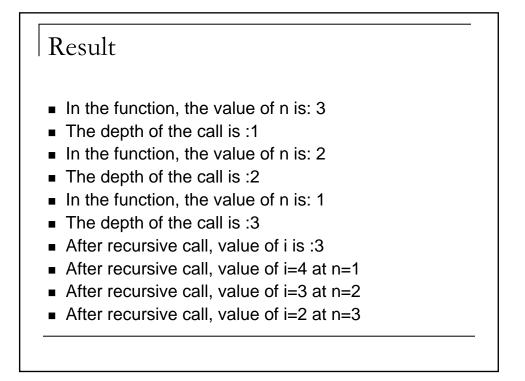
Illustration of recursion

```
#include<stdio.h>
void CountNum (int n)
{
   static int i = 1;
   printf("In the function, the value of n is:%d\n",n);
   printf("\n The depth of the call is %d\n",i++);
   if(n>1)
      CountNumber(n-1);
   printf("\n After recursive call, value of i=%d at
   n=%d\n", i--,n);
}
```





- When the CountNum function is called with N=1, the if statement is false.
- The function call CountNum is skipped.
- Thus the next statement is executed, which prints the argument value.
- At this stage the closing brace is encountered.
- The function does not return control to the main, but to the function body in CountNum, which gave the last call. The last call was in the if statement, hence the subsequent print statement is executed with i=3.
- This continues, till the final return to main.



Recursive Binary Search

- Searching and Sorting are inherently recursive in nature
- Many times, they follow a *divide and conquer* policy
- Binary Search searches a pre-sorted array of numbers
- In every function call, size of array to be sorted is halved
- Then, the half of the array in which the value might exist is checked
- If there is a possibility of finding the number in this half, this half is kept and the other half is discarded
- If there is no possibility of finding the number in this half, this half is discarded and the other half is considered

```
The Binary Search Code

#include<stdio.h>

#define MAX 20

int binsearch(int a[], int key, int low, int high)

{

    int pos, mid;

    mid=(low+high)/2;

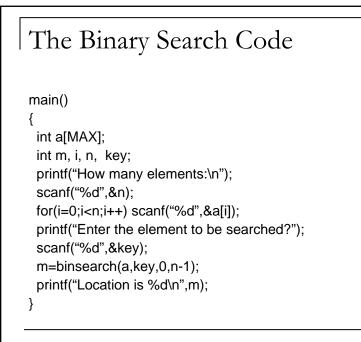
    if(a[mid]==key) return(mid);

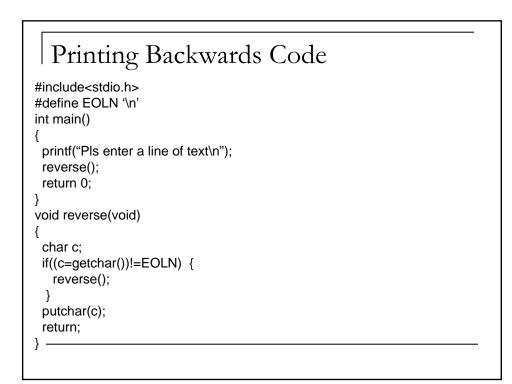
    else if(a[mid]>key) pos=binsearch(a,key,low,mid);

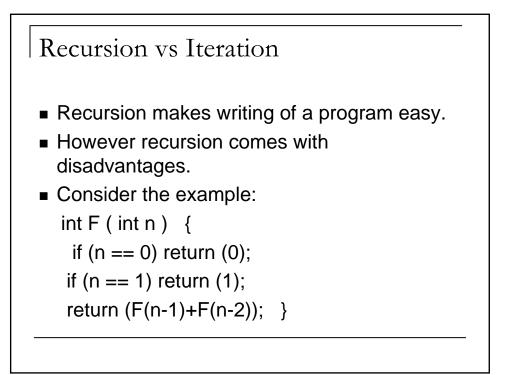
    else pos=binsearch(a,key,mid,high);

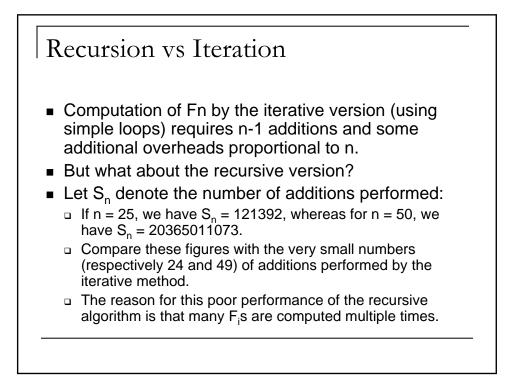
    return(pos);

}
```









Tail Recursion

- It is, therefore, often advisable to replace recursion by iteration.
- If some function makes only one recursive call and does nothing after the recursive call returns (except perhaps forwarding the value returned by the recursive call), then one calls this recursion a tail recursion.
- Tail recursions are easy to replace by loops: since no additional tasks are left after the call, no book-keeping need be performed, i.e., there is no harm if we simply replace the local variables and function arguments by the new values pertaining to the recursive call.
- This leads to an iterative version with the loop continuation condition dictated by the function arguments.

