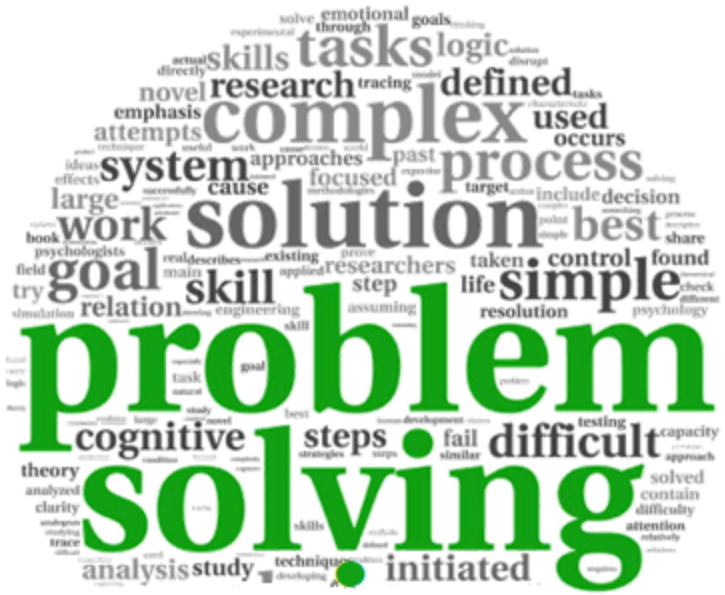


CSE 1051

19.2 INTRODUCTION TO RECURSION

Objectives:

To learn and understand the following concepts:

- ✓ To design a recursive algorithm
- ✓ To solve problems using recursion
- ✓ To understand the relationship and difference between recursion and iteration

Session outcome:

At the end of session one will be able to :

- Understand recursion
- Write simple programs using recursive functions

What is Recursion ?

- Sometimes, the best way to solve a problem is by solving a smaller version of the exact same problem first.

- Recursion is a technique that solves a problem by solving a smaller problem of the same type.

- A *recursive function is a function that invokes / calls itself* directly or indirectly.

- In general, code written recursively is shorter and a bit more elegant, once you know how to read it.

- It enables you to develop a natural, straightforward, simple solution to a problem that would otherwise be difficult to solve.

What is Recursion ?

- Mathematical problems that are recursive in nature like factorial, fibonacci, exponentiation, GCD, Tower of Hanoi, etc. can be easily implemented using recursion.

- Every time when we make a call to a function, a stack frame is allocated for that function call where all the local variables in the functions are stored. As recursion makes a function to call itself many times till the base condition is met, many stack frames are allocated and it consumes too much main memory and also makes the program run slower.

- We must use recursion only when it is easy and necessary to use, because it take more space and time compared to iterative approach.

Let us consider the code …

```
int main() {
  int i, n, sum=0;
  printf("Enter the limit“);
  scanf(“%d”,n);
  printf("The sum is %d“,fnSum(n));
  return 0;
}
```

```
int fnSum(int n)
{
   int sum=0;
   for(i=1;i<=n;i++)
                  sum=sum+i;
    return (sum);
}
```

Recursive Thinking .....

Recursion Process

A child couldn't sleep, so her mother told a story about a little frog,

who couldn't sleep, so the frog's mother told a story about a little bear,

who couldn't sleep, so bear's mother told a story about a little weasel

...who fell asleep.

...and the little bear fell asleep;

...and the little frog fell asleep;

...and the child fell asleep.

Steps to Design a Recursive Algorithm

- Base case:
  - It prevents the recursive algorithm from running forever.

- Recursive steps:
  - Identify the base case for the algorithm.
  - Call the same function recursively with the parameter having slightly modified value during each call.
  - This makes the algorithm move towards the base case and finally stop the recursion.

Let us consider same code again …

```
int main() {
   int i, n, sum=0;
   printf("Enter the limit“);
   scanf(“%d”, n);
   printf(“The sum is %d“,fnSum(n));
   return 0;
}
```

```
int fnSum(int x)
{
  if (x == 1) //base case
    return 1;
  else
    return x + fnSum(x-1) ;
                //recursive case
}
```

```
int fnSum(int n){
   int sum=0;
   for(i=1;i<=n;i++)
              sum=sum+i;
    return (sum);
}
```

Factorial of a natural number–

A classical recursive example

$$\mathbf{fact}(n) = \begin{cases} 1 & \text{if } n = 0 \\ n \cdot \mathbf{fact}(n-1) & \text{if } n > 0 \end{cases}$$

So factorial(5)

= 5* factorial(4)

= 4* factorial(3)

= 3*factorial(2)

= 2* factorial(1)

= 1*factorial(0)

= 1

Factorial- recursive procedure

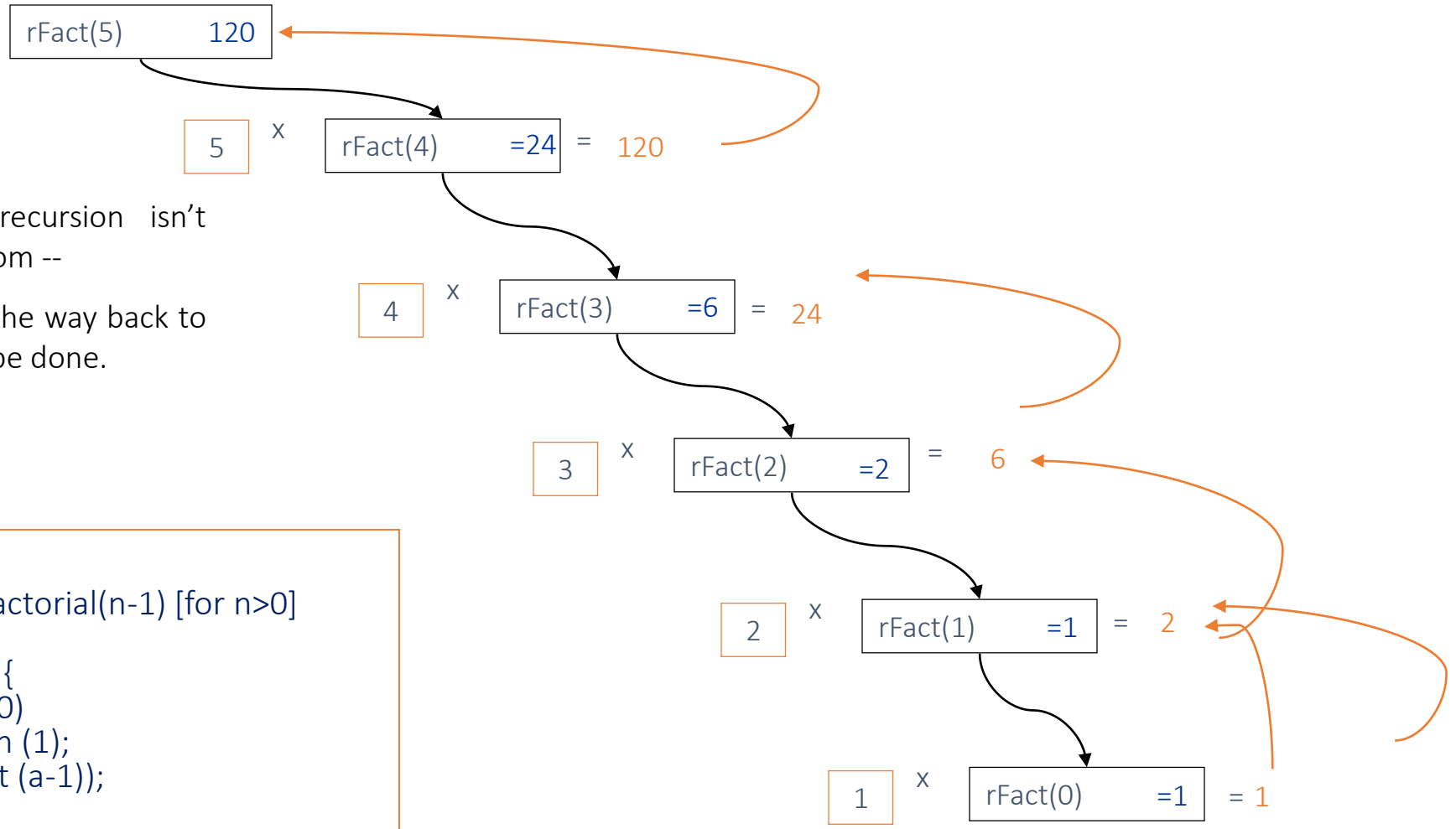
```
#include <stdio.h>

long factorial (long a) {
        if (a ==0) //base case
          return (1);
    return (a * factorial (a-1));
 }

int main () {

      long number;
      printf("Please type a number: “);
      scanf(“%ld”, &number);
      printf(“%ld ! = %d”,number,factorial (number));
      return 0;
     }
```

Recursion - How is it doing!

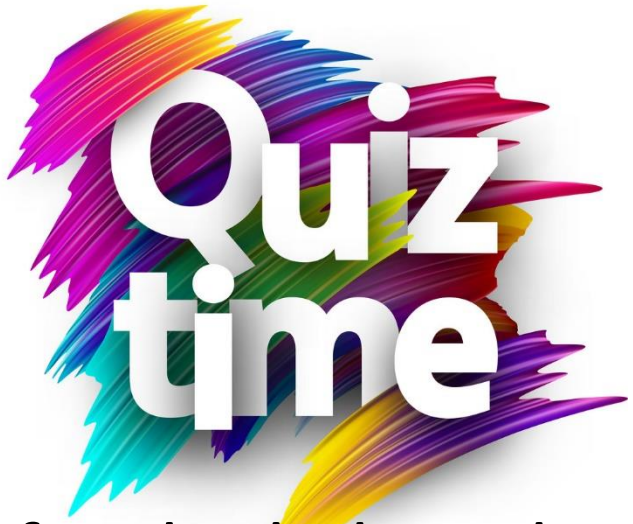


Notice that the recursion isn't finished at the bottom --

It must unwind all the way back to the top in order to be done.

factorial(0) = 1
factorial(n) = n * factorial(n-1) [for n>0]

```
long rFact (long a) {
                if (a ==0)
                  return (1);
   return (a * rFact (a-1));
 }
```



Go to posts/chat box for the link to the question **PQn. S19.2**

submit your solution in next 2 minutes

The session will resume in 3 minutes

Summary

- Recursion definition
- Example for recursive function