Chapter 2 - Recursion 1

- **Recursive Solutions**
  - A **recursive function** calls by itself.
  - Each recursive call solves an identical, but smaller, problem.
  - Test for base case enables recursive calls to stop.
  - Eventually, one of smaller problems must be the base case.

- **A recursive valued function: The factorial of n**

$$factorial(n) = n \times (n-1) \times (n-2) \times \cdots \times 1 \quad \text{for an integer } n > 0$$
$$factorial(0) = 1$$

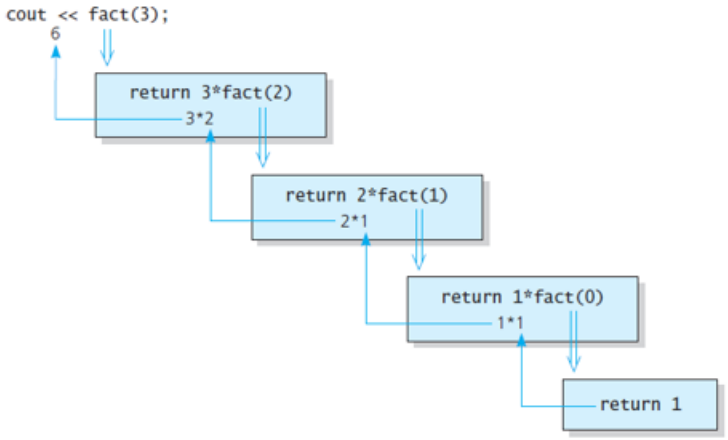
Note: Do not use recursion if a problem has a simple, efficient iterative solution

- A factorial solution

$$factorial(n) = \begin{cases} 1 & if\ n = 0 \\ n \times factorial(n-1) & if\ n > 0 \end{cases}$$

Trace the factorial function

n = 3: factorial (3) =



Question 1

Show how this function satisfies the properties of a recursive function. What's the return value of sumUpTo(n)? n >=1 integer.

```
int sumUpTo(int n)
{
    int sum = 0;
    if (n == 1)
        sum = 1;
    else
        sum = n + sumUpTo(n - 1);
    return sum;
}
```

A recursive void function: Writing a string backward

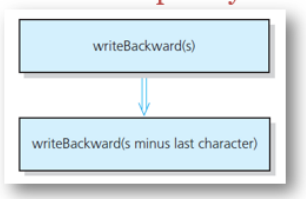
1. **Array based case**:

```
/** Writes the characters in an array backward.
  @pre  The array anArray contains size characters, where size >= 0.
  @post  None.
  @param anArray  The array to write backward.
  @param first  The index of the first character in the array.
  @param last  The index of the last character in the array. */
void writeArrayBackward(const char anArray[], int first, int last)
{
   if (first <= last)
   {
      // Write the last character
      cout << anArray[last];

      // Write the rest of the array backward
      writeArrayBackward(anArray, first, last - 1);
   }  // end if

   // first > last is the base case - do nothing
}  // end writeArrayBackward
```

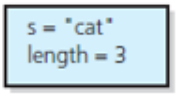
Ex.) char array[3] ={'c', 'a','t'};

2. **string**

Possible solution: strip away the last character



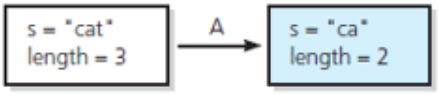
The initial call is made, and the function begins execution:



Output line: **t**

Point A (`writeBackward(s)`) is reached, and the recursive call is made.

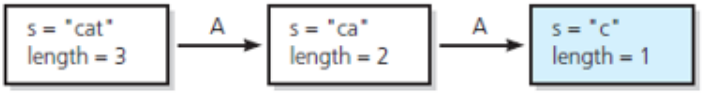
The new invocation begins execution:



Output line: **ta**

Point A is reached, and the recursive call is made.

The new invocation begins execution:



Output line: **tac**

```
string theString = "think";
theString.size();//returns the length of the string in theString
theString.substr(x, n);//returns a copy of a substring.
                       //The substring is n characters long and
                       //begins at position x of theString.

theString.size(); //returns 5
theString.substr(1, 2);//returns "hi"
```

Example)

```
string s = "cat";
s.size();
s.substr(0, 3);
s.substr(0, 1);
s.substr(0, 2);
```

```
void writeBackward(string s)
{
    int length = s.size();
    if (length > 0)
    {
        cout << s.substr(length - 1, 1);
        writeBackward(s.substr(0, length - 1));
    }
}
```

Quiz – 9/1/2021

Trace writeBackward(“think”). Show all steps.