Recursive Functions

Lecture 16
COP 3014 Spring 2017

February 13, 2017

Recursion

- Recursion is a mathematical technique that allows a calculation to be expressed in terms of itself.
- Ideal for "Lather, rinse, repeat" processes - procedures where "repeat the procedure" is part of the the procedure.
- In programming languages, this translates to having a function call itself.
- This is similar to a loop. However, a loop describes repetition in a iterative way, while recursion describes repetition in terms of itself.

Example

```
void recurse()
{
    recurse(); //Function calls itself
}

int main()
{
    recurse(); //Sets off the recursion
}
```

This will not run forever. The computer keeps function calls on a stack and once too many are called without ending, computer runs out of space and the program will crash.

It is important to write in a terminating condition while writing recursive functions.

Factorial through recursion

- A standard example for recursion is finding the factorial of a number.
- Factorial of a number is the product of all the numbers from 1 through the number.
- Factorials can also be defined as the number times the factorial of the previous number.
- The base, or stopping case, is when we reach 1 or 0. In that case, the factorial is 1, and we don't move on to the previous number.
- ```
  long factorial(long num)
  {
        if( num== 0 || num == 1 )
              return 1;
        return factorial(num-1)*num;
  }
  ```
  ```

Uses of recursion

- Recursion is very useful when dealing with the type of data that builds in on itself.
- Trees and graphs, which are very popular and efficient data structures use recursions.
- Be careful while using recursion. It is easy to write functions that don't terminate.