CS 211 Section 3: Recursion and Parsing

(by Nicholas Ruozzi and Lisa Minich)

Recursion is similar to induction in reverse. We want to take the k^{th} state and reduce it to the $(k-1)^{th}$ state. For example:

Summing up the numbers 0 through k can be done iteratively:

```
int sum =0;
for(int i=1; i< =k; i++)
{
 sum +=  i;
}
```

or recursively:

```
public int sum(int n)
{
  if(n==0)       //base case
    return 0;
  else           //recursive case
    return sum(n-1) + n;
}
```

Where the n^{th} state is added to the $(n-1)^{th}$ state for all n greater than the base case. Which in this example was 0.

Tail Recursion:

A method is tail recursive if the last action of the recursive method is the recursive call. For example, the solutions to problems 1 and 2 are not tail recursive, but the solution to problem 3 is tail recursive. Because recursion builds frame upon frame of the stack, the additional overhead required by non-tail recursive functions could be costly for large inputs.

Problem 1 (Reverse a string recursively):

```
public String reverseString(String word)
{
   if(word == null || word.equals(“”))
     return word;
   else
     return reverseString(word.substring(1, word.length())) + word.substring(0,1);
 }
```

Problem 2 (Remove consecutive duplicates from a string recursively):

For example, convert “aabccba” to “abcba”

```
public String removeDuplicates(String word)
{
   if(word == null || word.length() <= 1)
     return word;
   else if( word.charAt(0) == word.charAt(1) )
     return removeDuplicates(word.substring(1, word.length()));
   else
    return word.charAt(0) + removeDuplicates(word.substring(1, word.length()));
 }
```

Problem 3 (Compute n mod m without using %):

```
  public int modulus(int val, int divisor)
  {
     if(val < divisor)
       return val;
     else
       return modulus(val - divisor, divisor);
  }
```

Problem 1 (again) with Tail Recursion:

Someone asked in section if it was possible to convert non-tail recursive functions into tail recursive ones. The answer is usually yes. To illustrate how this can be done, I will rewrite problem one using tail recursion. To do this, I will utilize a helper function.

```
public String reverseString(String word)
{
      return tailReverse(word, “”);
 }

public String tailReverse(String word, String res)
{
   if(word == null || word.equals(“”))
     return res;
   else
     return tailReverse(word.substring(1, word.length()), res + word.charAt(0));
}
```

tailReverse is a tail recursive method that reverses a string.

Parsing:

All of the information on parsing can be found in the lecture notes.