**Recursion Activity**

**Example 1:**

int go( int num){

if( num < 0 )

return num; //Base Case

else

return num + go(num - 1); //Recursive Case

}

1. Locate the recursive case comment. Does it call the function go? What value is being sent as a parameter? **Yes, it calls itself, and it passes its own variable decreased by one.**
2. Locate the Base case comment. Does it call the function go? What condition is met that will allow the base case to run? **The base comment does not call the function itself. It instead check if the value passed into that instance is negative/less than 0.**
3. How many times will the function go be called if given a function call of... go( 4 ) **6 times: 4, 3, 2, 1, 0, -1, which will start returning up the chain.**

**Example 2**

void rain( int row ){

if( row <= 0 ) return;

else{

rain( row - 1 );

for( int col=0; col < row; col++)

cout << "# ";

cout << endl;

}

}

1. What is the recursive case in the above example? **If the row is 0 or higher it will call itself passing in the next value down: rain(row - 1);**
2. What will end the recursion? (base case) **If the value is 0 or less then it will return;**
3. Notice the coutstatement. Does it get displayed before or after the recursive call?

Notice the for loop above the coutstatement. What value is the number of hashtags dependent on? **The cout statement is called after the recursive call, meaning that they will be printed in ascending order, as the lower one will print then the next will return as they all come back. The value of the number of hashtags is reliant and the current call’s number that was passed in by the previous.**

1. Given a function call of rain( 4 ), what do you think will be displayed?
   1. **#**
   2. **##**
   3. **###**
   4. **####**
2. Run the code to see if you were correct**.**
3. How do you think the pattern would differ if the recursive call were after the cout statements? And why? **The order of the hashtags would be reversed, with 4 hashtags first going to 1. This would be because they are being printed each before the next/lower call is called, so 4 would print, then 3, then 2, then 1, then the loop would hit the base case and would end.**
4. Move the function call and check your response.

**Practice**

Complete the method so it will determine if the string is a palindrome. Use recursion instead of a loop.

#include <iostream>

using namespace std;

bool isPalindrome(string word, int first, int last ){

if (first == last || first+1 == last) {

return true;

}

if (word[first] != word[last]) {

return false;

}

isPalindrome(word, first+1, last-1);

return true;

}

*//Facilitate Recursive method call*

bool isPalindrome(string word){

return isPalindrome(word, 0, word.length()-1);

}

int main(){

string words[] = {"madam", "palindrome", "racecar"};

for(int i = 0; i < 3; i++){

cout << words[i] << ": "

<< (isPalindrome(words[i]) ? "Palindrome\n": "Not\n" );

}

return 0;

}