Jeremy Rios-Martinez

Comsc 265

Assignment 19

**Recursion Assignment**

1. **Research on the web. What are the 5 or 6 main types of recursion?**

LINEAR RECURSION - Recursion where only one call is made to the function from within the function (thus if we were to draw out the recursive calls, we would see a straight, or linear, path).

TAIL RECURSION - A recursive procedure where the recursive call is the last action to be taken by the function. Tail recursive functions are generally easy to transform into iterative functions.

BINARY RECURSION - A recursive function which calls itself twice during the course of its execution.

EXPONENTIAL RECURSION - Recursion where more than one call is made to the function from within itself. This leads to exponential growth in the number of recursive calls.

NESTED RECURSION - One of the arguments to the recursive function is the recursive function itself! These functions tend to grow extremely fast.

MUTUAL RECURSION - Functions that work in pairs or even larger groups. For example, function A calls function B which calls function C which in turn calls function A.

1. **Explain why recursion is like repetition with a ‘for’ loop or a ‘while’ loop.**

Because it repeatedly calls itself and executes the same code until some terminal condition is met, this is exactly what ‘for’ and ‘while’ loops do.

1. **Research the Tower Of Hanoi problem. Explain the Tower Of Hanoi game**

It is a [mathematical game](http://en.wikipedia.org/wiki/Mathematical_game) or [puzzle](http://en.wikipedia.org/wiki/Puzzle). It consists of three rods, and a number of disks of different sizes which can slide onto any rod. The puzzle starts with the disks in a neat stack in ascending order of size on one rod, the smallest at the top, thus making a conical shape.

The objective of the puzzle is to move the entire stack to another rod, obeying the following rules:

* Only one disk may be moved at a time.
* Each move consists of taking the upper disk from one of the rods and sliding it onto another rod, on top of the other disks that may already be present on that rod.
* No disk may be placed on top of a smaller disk.

**Find a recursive solution and get the code to work.**

#include "stdafx.h"

#include <iostream>

using namespace std;

void move(int n,char \*t1,char \*t2,char \*t3);

void main()

{

int n;

cout << "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

cout << "Sequence of steps to solve the Towers of Hanoi problem";

cout << "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n";

cout << "Enter the no. of rings ";

cin >> n;

move(n, "tower 1", "tower 2 ", "tower 3");

system("pause");

}

void move(int n,char \*t1,char \*t2,char \*t3)

{ // t1 stands for tower 1/source tower

if(n>0) // t2 stands for tower 2/intermediate tower

{ // t3 stands for tower 3/destination tower

move(n-1, t1, t3, t2);

// move n-1 disks from source to intermediate tower

cout << "disk " << n << " is moved from " << t1 << " to " << t3 << endl;

// move the disk from to source to destination

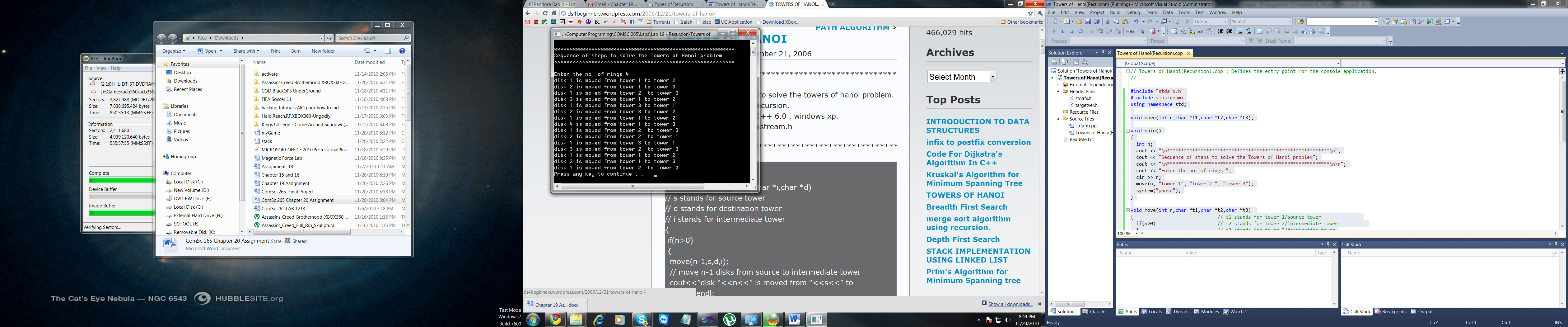
move(n-1, t2, t1, t3);

// move n-1 disks from intermediate to destination

}

}

**Print out the sequence of steps to solve a Tower with 4 rings.**



1. **When should you solve a problem recursively or non-recursively?**
2. Iterative functions are typically faster than their recursive counterparts. So, if speed is an issue, you would normally use iteration.
3. If the stack limit is too constraining then iteration is preferred over recursion.
4. Some procedures are very naturally programmed recursively.
5. **Create a flowchart for X to the power of N (See sections 19.5 and 19.6 in book)**

