**Advanced Java**

**Assignment 4**

**Recursion**

**100** **points**

**Name: Matthew Yackiel**  **Due: 10/14/2021 by 11:00 AM**

1. You do not need to complete all the items or do them in order. Include this sheet with your typed answers to the questions below.
2. What are the four basic principles of recursion? Explain each.

4 points:

1. Recursion requires a base case. The base case being the minimal version of the problem. It is where the recursive case stops.

2. The recursive case must progress towards the base case, and recursive step that does not progress towards the base case will repeat ad nauseam.

1. Compute the values of F(0), F(1), F(2), F(3) and F(50) for the function below. What mathematical function does F compute? 4 points How do we know this recursive method terminates? 4 points On separate paper **hand write** the recursive tree for F(5) and call stack for F(3). 10 points

public int F(int N) { return G(N,0,1);}

public int G(int N, int A, int B){

if(n<=0) return A;

return G(N-1, B, A + B);

}

1. Write and test recursive functions to do accomplish the following tasks. Write one project that will encompass the recursive code and have the main program test all routines.
   1. public int countOnes(int n); // return the number of ones in the binary representation of n. So if n=9, then the binary representation of n is 1001 and the number of ones is 2. **10 points**

Good test program: **3 points**

Type the recursive question here or how does the recursion work? **2 points**

Converting to binary is the same as taking the integer mod 2 and then dividing by 2 and outputting the mod 2 calculation form left to right. In this case we don’t care about the order we just want to count the ones. So dividing by two until we arrive at 0 while adding 1 to our count every time we encounter a remainder of

1. We repeat until we reach zero at which point we return 0.

* 1. public String binaryRepresentation(int n); //return the binary representation of n. **10 points**

Good test program: **3 points**

Type the recursive question here or how does the recursion work? **2 points**

This problem differs from the previous one in that it cares about the order

of 0’s and 1’s as opposed to only caring about the amount of 1’s in the binary.

We want to know when we reach 1 this time around as we can append a 1 to the string. The recursive case is still extremely similar.

* 1. public long digitSum(int n); // return the sum of the digits in n

**10 points**

Good test program: **3 points**

Type the recursive question here or how does the recursion work. **2 points**

This one is even more simple than the countOnes method. We divide by ten until we get to an integer value of zero, we add n % 10 to the total since n%10 is the ones place digit of n at every recursive step.

* 1. public long power(int x, int n); // Compute x raised to the nth power (xn)
     1. Use the following idea x0 is 1. x-n is 1.0 / xn . If n is positive and even then xn  = (xn/2 )2  if n is positive and odd then xn  = xn-1\* x.

**5 points**

Good test program: **3 points**

* 1. Write a method to compute the number of combinations of N elements taken K at a time, C(N,K), using the recursive idea suggested by Pascal’s triangle explained on the next page. You must also create good test program.

public long C(int N, int K) ; // number of combinations of N elements taken K at a time. **25 points**

