Alexander Shah
Homework 3: Recursion
EN.605.202.81 Section 84

1.
We can calculate a sum of two nonnegative integers by recursively calling our sum function and establishing a base case. With each recursive call, the value of b is decremented and the value of a is incremented. Our base case is when b is 0, when there is no more to add to a.

```
Method recursiveSum(a, b)
   if b==0
      return a
   else
      return recursiveSum(a+1, b-1)
```

2.
In this recursive algorithm, we pass along A, the array of integers, and n, the position in the array we are, initially set to the length of the list. We recursively call our method with the previous element in the list. When we reach our base case when n==1, we return the first element of the list, and get the sum of the elements from the beginning of the array to n-2, we then add the element at n-1 and divide by n to get the average of all the elements.

```
Method arrayAvg(A, n) //where A is the array and n is the length of the array A
   if n==1
      then return A[n-1]
   else
      return (arrayAvg(A, n-1)*(n-1) + A[n-1])/n
```

3.
Each time we recursively call binary search we halve the search space, so at worst the algorithm could halve the search space until there is only one element left. This depends on the number of elements n, and would occur within log(n) +1 calls, log(n) to search the space recursively by dividing it repeatedly, and +1 to account for the initial call.

4.
We are given the cases if y divides evenly by x, if x < y, and when y > x. Our base case is when y divides evenly by x, our recursive cases are when they are unequal. If x< y we reverse the order of the parameters, and recursively call our function again. We are trying to to take one “y” out of our x in the case y > x, in which we recursively call the gcd on (y, x%y) which reduces the value of the right parameter. Eventually we will whittle down the x and y values until we can return a value in the case they divide evenly.

```
Method gcd(x, y)
   if x % y==0
      return y
   if x < y
      gcd(y, x)
   else
      gcd(y, x%y)
```

5.
Recursive fibonacci is defined with two base cases for values 0 and 1, and a recursive case when n>1. The recursive case involves the addition of the two previous fib elements, giving us the fibonacci sequence. Gfib involves passing starting parameters f0 and f1.

```
Method gfib(f0, f1, n)
   if n==0
      return f0
   if n==1
      return f1
   else
      return gfib(f0, f1, n-1) + gfib(f0, f1, n-2)
```