Assignment 3 – Recursion

1.

```
Int add(int a, int b)
If (a == 0){
Return b;
Else if
Return 1+ add(b, a-1);
```

2.

Recursive algorithm for average:

```
double average(int a[], int b)
double total = a[]/b.length;
If( b == 1)
double total = a[0];
Else if
total = a[b-1] + (b-1) * average(a, b-1);
double Result = total/b
return result;
```

3.

The maximum number of recursive calls made by the binary search algorithm would be log n.
Lets say we take an array like this

1	2	3	4	5	6	7	8	9

If we wanted to search this array for the element number 3 then we would take the mid value which is taken by doing (0+8)/2 which would be the 4 index . since the 4^{th} index isn't equal to 5 then we would take the next portion of the array to the left and divide by two which would give us the index 2 which is equal to the element of 3. This would be 2 recursive calls.

4.

```
Int gcd(int x, int y)
If ((y<= x) && (x%y = 0)
Then result is = y;
Else if  x< y then
Return the gcd(y, x)
else
return gcd(y, x%y)
```

5.

```
Int fib(int f0, int f1, int n)
If (n== 0)
Then return result = f0
Else if  (n== 1)
Then return= f1
Otherwise return =gfib(f0,f1,n-1) + gfib(f0,f1,n-2)
```

6.

```
Acker(int m, int n)
If m =0
```

Then return n+1
Else if (n=0)
Return a(m-1,1)
Otherwise
Return a(m-1,a(m, n-1))

A(2,2 = a(1, a(2,1))
= a(1,a(1,a(2,0)))
= a(1,A(1,A(1,1)))
= A(1,A(1,A(0,A(1,0))))
= a(1, a(1,a(0,a(0,1))))
= a(1,a(1,a(0,2)))
=a(1,a(1,3))
= a(1,a(0,a(1,2)))
= a(1,a(0,a(0,a(1,1))
= a(1,a(0,a(0,a(0,a(1,0)))))
=a(1,a(0,a(0,a(0,a(0,1))))))
= a(1,a(0,a(0,a(0,2))))
= a(1,a(0,a(0,3)))
= a(1,a(0,4))
= a(1, 5)
=a(0,a(1,4))
= a(0,a(0,a(1, 3)))
= a(0,a(0,a(0,a(1,2))))
= a(0,a(0,a(0,a(0,a(1,1)))))
= a(0,a(0,a(0,a(0,a(0,a(1,0)))))
= a(0,a(0,a(0,a(0,a(0,a(0,1))))
= a(0,a(0,a(0,a(0,a(0,2))))
= a(0,a(0,a(0,a(0,3))))
= a(0,a(0,a(0,4)))
= a(0,a(0,5))
=a(0,6)
= 7

7.

```
int rec(int n)
{
   if ( f(n) == FALSE ) {
      /* any group of statements that  do not change the value of n */
      return (rec(g(n)));
   }//end if
   return (0);
}//end rec
```

Int iterative(int n)
If (f(n) == true)
Return 0
If (f(n) == false)
N=(g(n))

Return 0