Assignment 3 – Recursion

Write pseudo-code not Java for problems requiring code. You are responsible for the appropriate level of detail.

Q1 and Q2 are intended to help you get comfortable with recursion by thinking about something familiar in a recursive manner. Q3 – Q6 are practice in working with non-trivial recursive functions. Q7 and Q8 deal with the idea of conversion between iteration and recursion.

1. Write a recursive algorithm to compute *a*+*b*, where *a* and *b* are nonnegative integers.

ANSWER:

```
int add(int a, int b) {
   if (b==0) {
      return a;
   }
   else {
      return 1 + add(a, b-1);
   }
}
```

2. Let A be an array of integers. Write a recursive algorithm to compute the average of the elements of the array. Solutions calculating the sum recursively, instead of the average, are worth fewer points.

ANSWER:

```
double getArrayAverage(int[] A) {
   double total;
   double average;
   int index;
   if (index==0) { // will be 0 when initialized
      total = A[0]; // initialize total to first element in array
   }
   else {
      total = A[index] + (index+1)*getArrayAverage(A);
   }
   index++;
   average = total/index;
   return average;
}
```

3. If an array contains n elements, what is the maximum number of recursive calls made by the binary search algorithm?

ANSWER:

log(n) (rounded up)

Example:
[1, 2, 3, 4, 5, 6, 7, 8, 9] Looking for index 1
Pick 5, choose lower half of array
[1, 2, 3, 4]
Pick 3, choose lower half of array
[1, 2]
Pick 2, choose lower half of array
[1]
Found it in 4 iterations
log(9) rounded up = 4

4. The expression m % n yields the remainder of m upon (integer) division by n. Define the greatest common divisor (GCD) of two integers x and y by:

gcd(x, y) = y	**if (y ≤ x and x%y == 0)**
gcd(x, y) = gcd(y, x)	**if (x < y)**
gcd(x, y) = gcd(y, x%y)	**otherwise**

Write a recursive method to compute gcd(x,y).

```
int gcd(int x, int y) {
   if(x==y) {
      return y;
   }
   else if(x < y) {
      return gcd(x, y);
   }
   else {
      return gcd(x, y%x);
   }
}
```

5. A generalized Fibonacci function is like the standard Fibonacci function,, except that the starting points are passed in as parameters. Define the generalized Fibonacci sequence of f0 and f1 as the sequence gfib(f0, f1, 0), gfib(f0, f1, 1), gfib(f0, f1, 2), ..., where

gfib(f0, f1, 0) = f0
gfib(f0, f1, 1) = f1
gfib(f0, f1, n) = gfib(f0, f1, n-1) + gfib(f0, f1, n-2) if n> 1

Write a recursive method to compute gfib(f0,f1,n).

```
int gfib(int f0, int f1, int n) {
   if(n==0) {
      return f0;
   }
   else if(n==1) {
      return f1;
   }
   else {
```

```
        return gfib(f0, f1, n-1) + gfib(f0, f1, n-2);
    }
}
```

6. Ackerman's function is defined recursively on the nonnegative integers as follows:

$a(m, n) = n + 1$ if $m = 0$
$a(m, n) = a(m-1, 1)$ if $m \neq 0, n = 0$
$a(m, n) = a(m-1, a(m, n-1))$ if $m \neq 0, n \neq 0$

Using the above definition, show that a(2,2) equals 7.

a(2, 2)
= a(2-1, a(2, 1))
= 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. Convert the following recursive program scheme into an iterative version that does not use a stack. *f(n)* is a method that returns TRUE or FALSE based on the value of n, and *g(n)* is a method that returns a value of the same type as *n* (without modifying *n*).

```
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
```

ANSWER

```
int rec(int n) {
    while(f(n) == false
   {
        n = g(n);
    }
    return(0);
}
```