CS526 O2
Homework Assignment 2

Problem 1 (20 points). This is practice for analyzing running time of algorithms. Express the running time of the following methods, which are written in pseudocode, using the *big-oh* notation. Assume that all variables are appropriately declared. You must justify your answers. If you show only answers, you will not get any credit even if they are correct.

(1)

```
method1(int[ ] a)  // returns integer
 x = 0;
 y = 0;
 for (i=1; i<n; i++) { // n is the number of elements in array a
             if (a[i] == a[i-1]) {
                x = x + 1;
             }
             else {
                y = y + 1;
             }
      }
      return (x - y);
```

Answer: The for loop of line 4 takes O(n). Other lines only are only concerned about variable assignments and integer operations, which take constant amount of time. So the running time of this algorithm is O(n) as we discard the lower order term and ignore the coefficient.

(2)

```
method2(int[ ] a, int[ ] b)  // assume equal-length arrays
      x = 0;
      i = 0;
      while (i < n) { // n is the number of elements in each array
            y = 0;
           j = 0;
            while (j < n) {
                  k = 0
                  while (k <= j) {
                        y = y + a[k];
                        k = k + 1;
                  }
                  j = j + 1;
            }
            if (b[i] == y) {
                  x++;
            } i = i +
            1;
      }
      return x;
```

Answer: The while loops of line 4, 7, 9 all take O(n) each, so the nested while loops take $O(n^3)$. Other lines only are only concerned about variable assignments and integer operations, which take constant amount of time C. So the running time of this algorithm is $O(n^3)$ as we discard the lower order term and ignore the coefficient.

(3)

```
// n is the length of array a
// p is an array of integers of length 2
// initial call: method3(a, n-1, p)
// initially p[0] = 0, p[1] = 0
method3(int[] a, int i, int[] p)
    if (i == 0) {
            p[0] = a[0];
            p[1] = a[0];
    }
    else {
            method3(a, i-1, p);
            if (a[i] < p[0]]) {
                    p[0] = a[i];
            }
            if (a[i] > p[1]]) {
                    p[1] = a[i];
            }

    }
```

Answer: Each method3 invocation takes a constant amount of time, and method3 is invoked n times, so it takes O(n). Other lines only are only concerned about variable assignments which take constant amount of time. So the running time of this algorithm is O(n) as we discard the lower order term and ignore the coefficient.

(4)

```
// initial call: method4(a, 0, n-1) // n is the length of array a
public static int method4(int[] a, int x, int y)
{
   if (x >= y)
   {
        return a[x];
   }
   else
   {
           z = (x + y) / 2; // integer division
           u = method4(a, x, z);
           v = method4(a, z+1, y);
```

```
            if (u < v) return u;
            else return v;
        }
    }
```

Answer: Each method4 invocation takes a constant amount of time, and method4 is invoked at most (log n + 1) times by u and (log n + 1) times by v, so it takes O(log n) for the recursions. Other lines only are only concerned about variable assignments which take constant amount of time. So the running time of this algorithm is O(log n) as we discard the lower order term and ignore the coefficient.

Problem 2 (20 points) This problem is about the stack and the queue data structures that are described in the textbook.

(1) Suppose that you execute the following sequence of operations on an initially empty stack. Using Example 6.3 in the textbook as a model, complete the following table.

Operation	Return Value	Stack Contents
push(10)	-	(10)
pop( )	10	( )
push(12)	-	(12)
push(20)	-	(12, 20)
size( )	2	(12, 20)
push(7)	-	(12, 20, 7)
pop( )	7	(12, 20)
top( )	20	(12, 20)
pop( )	20	(12)
pop( )	12	( )
push(35)	-	(35)
isEmpty( )	false	(35)

(2) Suppose that you execute the following sequence of operations on an initially empty queue. Using Example 6.4 in the textbook as a model, complete the following table.

Operation	Return Value	Queue Contents (first ← Q ← last)
enqueue(7)	-	(7)
dequeue( )	7	( )
enqueue(15)	-	(15)

enqueue(3)	-	(15, 3)
first( )	15	(15, 3)
dequeue( )	15	(3)
dequeue( )	3	( )
first( )	null	( )
enqueue(11)	-	(11)
dequeue( )	11	( )
isEmpty( )	true	( )
enqueue(5)	-	(5)