CS526 O2
Homework Assignment 2

Problem 1 (20 points). This problem is a practice of analyzing running time of algorithms. Express the running time of the following methods, which are written in a pseudocode style, 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 though 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);
```

(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;
```

(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];
          }

    }
```

(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;
      }
    }
```

Problem 2 (20 points). This problem is a practice of drawing recursion traces of recursive algorithms.

(1) Draw the recursion trace for the computation of power(2, 5) using the algorithm of Code Fragment 5.8 in the textbook (page 209)

(2) Draw the recursion trace for the computation of power(2, 13) using the algorithm of Code Fragment 5.9 in the textbook (page 210)

Examples of recursion traces are shown in Figure 5.10 and Figure 5.12 in the textbook. You may want to use these examples as models when you draw recursion traces.

Problem 3 (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)		
pop( )		

push(12)		
push(20)		
size( )		
push(7)		
pop( )		
top( )		
pop( )		
pop( )		
push(35)		
isEmpty( )		

(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)		
dequeue( )		
enqueue(15)		
enqueue(3)		
first( )		
dequeue( )		
dequeue( )		
first( )		
enqueue(11)		
dequeue( )		
isEmpty( )		
enqueue(5)		

Problem 4 (40 points) The goal of this problem is: (1) practice of using and manipulating a doubly linked list and (2) practice of designing and implementing a small recursive method. Write a program named *Hw2_p4.java* that implements the following requirement:

- This method receives a doubly linked list that stores integers.
- It reverses order of all integers in the list.
- This must be done a ***recursive*** manner.
- The signature of the method must be:

```
public static void reverse(DoublyLinkedList<Integer> intList)
```

- You may want to write a separate method with additional parameters (refer to page 214 of the textbook).
- You may not use additional storage, such as another linked list, arrays, stacks, or queues. The rearrangement of integers must occur within the given linked list.

An incomplete *Hw2_p4.java* is provided. You must complete this program.

You must use the *DoublyLinkedList* class that is posted along with this assignment. You must not use the *DoublyLinkedList* class that is included in textbook's source code collection.

Note that you must not modify the given *DoublyLinkedList* class and the *DoubleLinkNode* class.

Deliverable

No separate documentation is needed for the program problem. However, you must include the following in your source code:

- Include the following comments above each method:
  - Brief description of the method
  - Input arguments
  - Output arguments
- Include inline comments within your source code to increase readability of your code and to help readers better understand your code.

You must submit the following files:

- *Hw2_p1_p3.pdf*. This file must include the answers to problems 1, 2, and 3.
- *Hw2_p4.java*

Combine all files into a single archive file and name it *LastName_FirstName_hw2.EXT*, where *EXT* is an appropriate archive file extension, such as *zip* or *rar*.

Grading

Problem 1 (20 points):

- Up to 4 points will be deducted for each wrong answer.

Problem 2-(1) (10 points):

- Up to 8 points will be deducted if your answer is wrong.

Problem 2-(2) (10 points):

- Up to 8 points will be deducted if your answer is wrong.

Problem 3-(1) (10 points):

- Up to 8 points will be deducted if your answer is wrong.

Problem 3-(2) (10 points):

- Up to 8 points will be deducted if your answer is wrong.

Problem 4 (40 points):

- If your program does not compile, 32 points are deducted.
- If your program compiles but causes a runtime error, 24 points are deducted.
- If there is no output or output is completely wrong, 20 points are deducted.
- If your program is partly wrong, up to 20 points are deducted