Points off

1	2	3	4A	4B	4C	5A	5B	Total Off	Net

CS 307 – Final – Spring 2011

Name____

UTEID login name ______________________________

Instructions:

1. Please turn off your cell phones and all other electronic devices.
2. There are 5 questions on this test.
3. You have 3 hours to complete the test.
4. You may not use a calculator on the test.
5. You may add helper methods if you wish when answering coding questions.
6. When answering coding questions assume the preconditions of the methods are met.

1. (1.5 point each, 30 points total) Short answer. Place you answers on the attached answer sheet.

For questions that ask "what is the output":

- If the code contains a syntax error or other compile error, answer "Compiler error".
- If the code would result in a runtime error or exception answer "Runtime error".
- If the code results in an infinite loop answer "Infinite loop".

On questions that ask for the order (Big O) of a method or algorithm, recall that when asked for Big O your answer should be the most restrictive, correct Big O function. For example Selection Sort is order O(N^2), but per the formal definition of Big O it is correct to say Selection Sort is O(N^3), O(N^4), O(2^N), and O(N!). Give the most restrictive, correct function. (Closest without going under.)

A. The following values are inserted one at a time in the order shown into an initially empty binary search tree using the traditional, naïve algorithm. Draw the resulting tree.

```
12   17   -5   13   21   0   -5
```

B. What is the result of the following postfix expression?

```
2 3 4 6 + + *
```

Consider the following binary tree. `37` is the root of the tree.

```
         37
        /  \
      20    25
     /  \     \
   12    19   23
              /
            21
```

C. What is the result of a pre-order traversal of the binary tree shown above?

D. What is the result of a in-order traversal of the binary tree shown above?

E. What is the result of a post-order traversal of the binary tree shown above?

F. Is the tree shown above a binary search tree?

G. Recall the `UnsortedSet` class you implemented on Assignment 8 that used an ArrayList as its internal storage container. What is the order (big O) of inserting N distinct values (no duplicates) in random order into the `UnsortedSet` using the add method.

H. Consider the following method:

```
public int total_h(ArrayList<Integer> list) {
     int total = 0;
     for(int i = 0; i < list.size(); i++)
          total += list.get(i);
     return total;
}
```

It takes 0.5 seconds for the above method to complete when `list.size()` = 1,000,000. What is the expected time for the method to complete when `list.size()` = 3,000,000?

I. A sorting method implements the quicksort algorithm to sort an array of `int`s. It takes the method 4 seconds to complete when sorting an array with 1,000,000 distinct `int`s in random order. What is the expected time for the method to complete when sorting an array with 4,000,000 distinct `int`s in random order?

J. What is output by the following code? (The code does **not** contain a syntax error.)

```
Queue<Integer> q = new Queue<Integer>();
int val = 1;
for(int i = 0; i < 6; i++) {
     val = val + i;
     q.enqueue(val);
}

for(int i = 0; i < q.size(); i++)
     System.out.print(q.dequeue() + " ");
```

K. What is output by the following code?

```
int[] data = {12, 15, 4, 7, 11, 6, 9};
Stack<Integer> st = new Stack<Integer>();
for(int x : data) {
     if(x % 3 == 0)
          st.push(x);
     else
          st.push(x / 10);
}

while(!st.isEmpty()) {
     int x = st.pop();
     if(!st.isEmpty() && x < st.top())
          System.out.print(x + " ");
}
```

L. What is the worst case order (Big O) for inserting N distinct elements into an initially empty Red-Black Tree? The elements are inserted one at a time.

M. What is the best case order (Big O) for inserting N distinct elements into an initially empty Binary Search Tree that uses the traditional, naïve insertion algorithm? The elements are inserted one at a time.

N. You want to encode 200 distinct colors with as few bits as possible. What is the minimum number of bits necessary to encode the 200 distinct colors?

O. 1,000,000 distinct elements in random order are inserted into an initially empty Binary Search Tree that uses the traditional, naïve insertion algorithm. What is the expected height of the resulting tree? State the **actual** expected height, not the order of the height.

P. 500 binary searches are performed on an array with 1000 `int`s in ascending order. In total, how many total elements do you expect to be examined when performing the 500 binary searches?

Q. In one sentence explain why the following code will not compile.

```
ArrayList<Integer> list1 = new ArrayList<Integer>();
String name = "Isabelle";
list1.add(name.length());
list1.add(name);
list1.add( (int) name.charAt(2) );
```

R. In one sentence explain why the following code will not compile. (The code uses the Java `LinkedList` class.)

```
LinkedList list2 = new LinkedList();
list2.add(12);
list2.add("Olivia");
list2.add( new ArrayList<Integer> () );
System.out.println( list2.get(1).substring(2, 5) );
```

S. In one sentence explain why the following class will not compile.

```
public class Elem implements Comparable {

     private int val;

     public Elem(int v) { val = v; }

     public void inc() { val++; }

}
```

T. What is output by the following code?

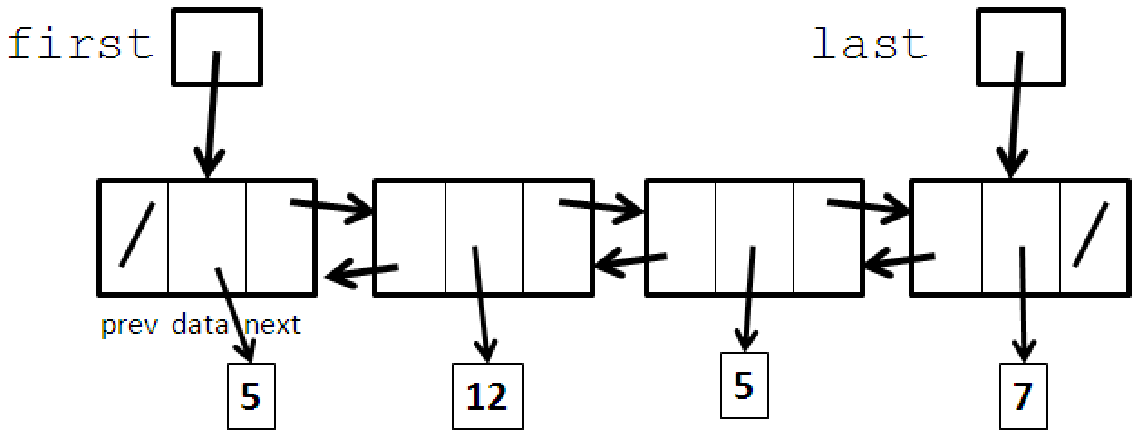
```
TreeSet<Character> ts = new TreeSet<Character>();
String word = "ONOMONOPIA";
for(int i = 0; i < word.length(); i++)
     ts.add(word.charAt(i));

for(char ch : ts)
     System.out.print(ch);
```

2. (Linked Lists, 15 points) Complete an instance method that clears a LinkedList and meets the requirements below.

The linked list has the following properties:

- The nodes are doubly linked with references to the next node in the list and the previous node in the list.
- The `LinkedList` class maintains references to the first and last nodes in the list. No dummy or header node is used. If the list is empty, the references to the first and last nodes in the list are set to `null`.
- The first node in the list has its previous reference set to `null` and the last node in the list has its next reference set to `null`.
- The `LinkedList` class does not have an instance variable for its size.
- The `LinkedList` class achieves genericty by using Java's generics.

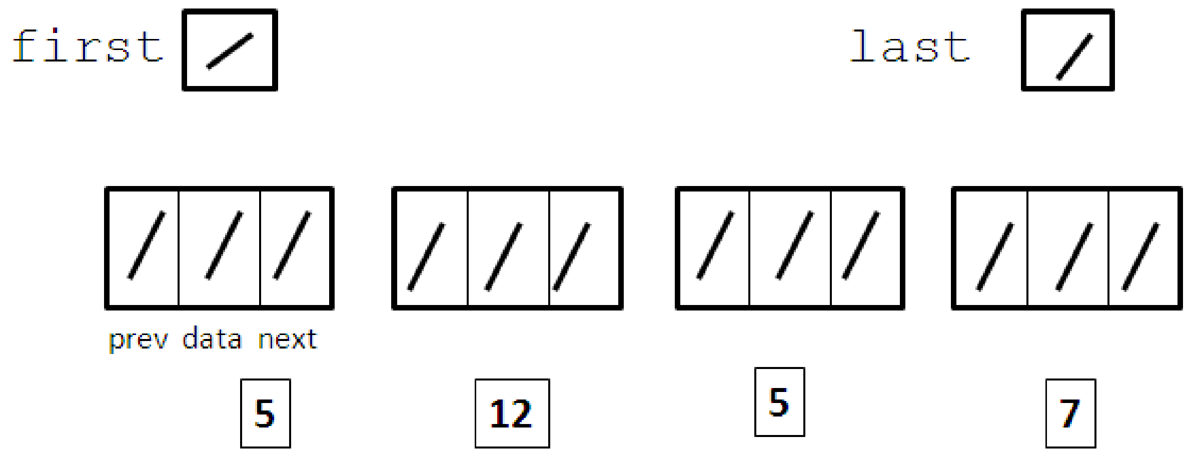
The structure of the internal storage container for a `LinkedList` that represents the list `[5, 12, 5, 7]` is shown below. Arrows indicate the references stored by variables. Forward slashes indicate references that store `null`.



The clear method for the `LinkedList` class could be written in the following way:

```
public void clear() {
     first = null;
     last = null;
     // let the garbage collector do its job
}
```

However, you want to run a test though to see if setting all references of all nodes in the linked list to `null` allows the garbage collector to run faster. Your clear method must manually set all references in all nodes to `null`. For example, if the `clear` method was called on the `LinkedList` shown above, the structure would be altered as shown in the image at the top of the next page. (Forward slashes indicate references that store `null`.)



The `LinkedList` class uses the following node class:

```
public class Node<E>{
    public Node(Node<E>, prev, E value, Node<E> next)

    public Node<E> getNext()
    public Node<E> getPrev()
    public E getValue()

    public void setNext(Node<E> newNext)
    public void setPrev(Node<E> newPrev)
    public void setValue(E obj)
}
```

Here is the `LinkedList` class.

```
public class LinkedList<E>{

     private Node<E> first;
     private Node<E> last;
```

Your method must be O(1) *space*, meaning you cannot use temporary arrays, lists, or other data structures whose size depends on the number of elements in the linked list.

Complete the following instance method for the `LinkedList` class:

```
// pre: none
// post: per the question description
public void clear() {
```

Complete this method on the next page.

Complete the following instance method for the `LinkedList` class:

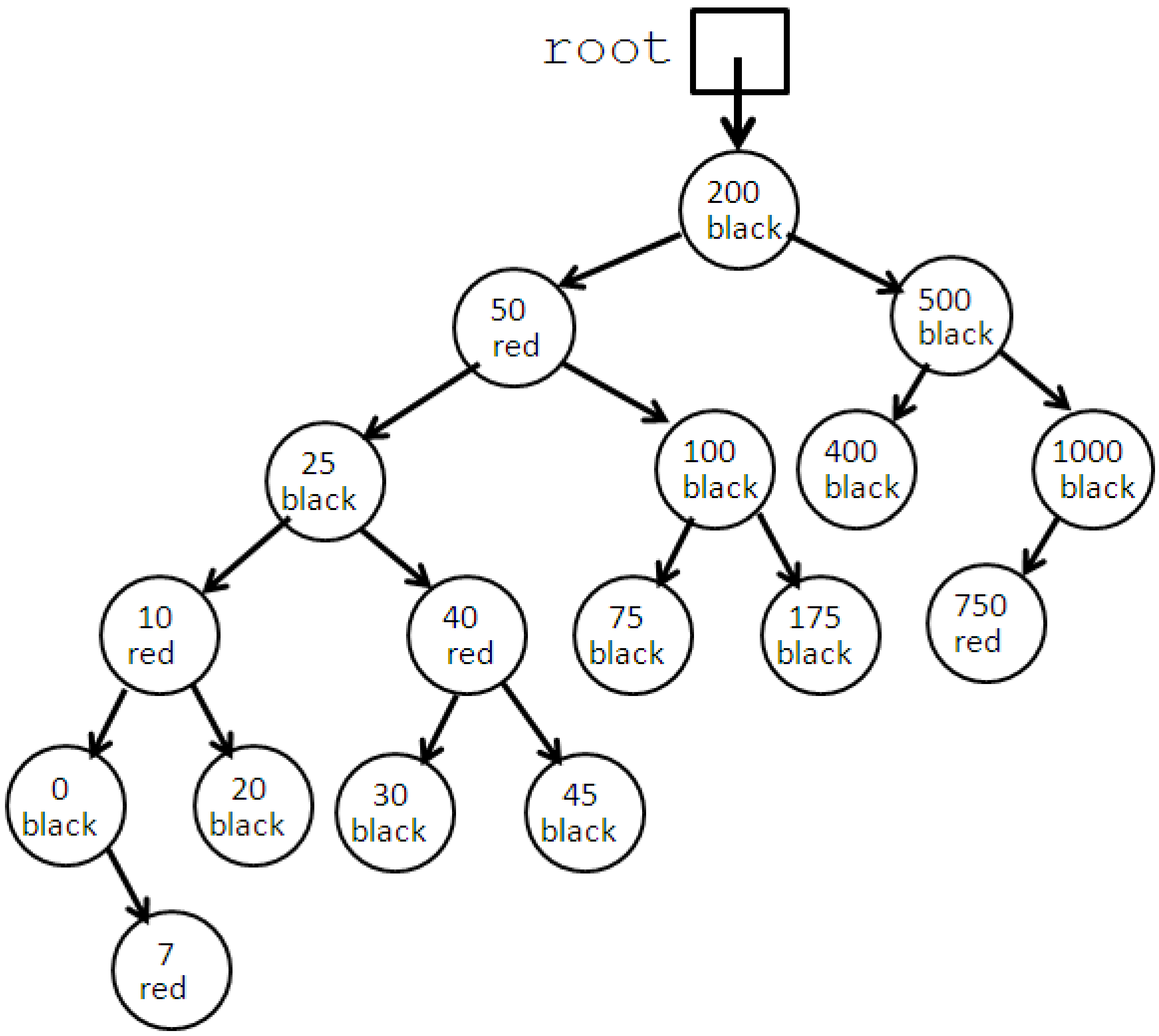
```
// pre: none
// post: per the question description
public void clear() {
```

3. (Binary search trees, 25 points) This question has three parts. You will write methods to verify if some of the requirements of a Red Black tree are met.

Recall a Red Black tree is a binary search tree with added requirements:

1. every node is colored red or black
2. the root is colored black
3. if a node is colored red it cannot have any children also colored red
4. all paths from the root of the tree to null links contain the same number of black nodes.

Consider the following *potential* red black tree:



The `BST` class used for this question is:

```
public class BST<E extends Comparable<? super E>> {

     private RBNode<E> root; // when tree is empty, root == null
```

Part A (5 points): Write a private method in the `BST` class that determines the number of black nodes in the path from the root of the tree to the left-most node in the tree. In the tree shown on the previous page the method would return 3. (The path from the root of the tree to the left most node contains the values 200, 50, 25, 10, 0 and 3 of those nodes are colored black.. **You may not use any classes in this question other than the `RBNode` class shown below.**

The `BST` class uses the following node class:

```
public class RBNode<E extends Comparable<? super E>> {

     // create a new node with the given values
     public RBNode (E value, RBNode <E> left,
                    RBNode <E> right, boolean isBlack)

     public E getValue()
     public RBNode <E> getLeft()
     public RBNode <E> getRight()
     public boolean isBlack() /* returns true for nodes colored
                                black, false for nodes colored red */

     public void setValue(E newValue)
     public void setLeft(RBNode <E> newLeft)
     public void setRight(RBNode <E> newRight)
     public void setIsBlack(boolean isBlack)
}
```

Complete the following instance method for the `BST` class:

```
// pre: none
// post: return the number of black nodes in the path from the
//       root of this tree to the left-most node.
private int numBlackNodesFromRootToLeftMost() {
```

Part B (8 points) Complete an instance method for the `BST` class that returns true if all paths from the root of the tree to null links contain the same number of number of black nodes or false if they do not. Given the tree shown on page 8, the method would return true.

The method is kicked off by the following method:

```
private boolean allPathsCorrect() {
     int magicNumber = numBlackNodesFromRootToLeftMost();
     return allPathsCorrectHelper(root, magicNumber, 0);
}
```

You may not use any classes in this question other than the `RBNode` class from part A.

Complete the following instance method in the `BST` class:

```
private boolean allPathsCorrectHelper(RBNode<E> n, int magicNum,
                              int blackNodesInCurrentPath) {
```

Part C (12 points). Complete an instance method for the `BST` class that returns true if the red requirement (number 3 from the list on page 8) of a red black tree is met. In other words if there are no red nodes in the tree with red children, the method returns true, otherwise it returns false. Given the tree shown on page 8, the method would return true.

The method is kicked off by the following method:

```
private boolean redRuleMet() {
     return redHelper(root);
}
```

You may not use any classes in this question other than the `RBNode` class from part A.

Complete the following instance method in the `BST` class:

```
private boolean redHelper(RBNode<E> n) {
```

4. (Using Data Structures. 15 points) Complete a method that determines if the tags in an XML file are properly formatted and nested.

XML is a set of rules for structuring data in a way that can be decoded programmatically. For this question assume XML files contain data and tags. There are two types of tags, opening tags and closing tags. Opening tags start with a `'<'` character and end with a `'>'` character. Opening tags contain a non empty string that does not contain any `'<'` or `'>'` characters. Closing tags start with the string `"</"` and end with the `'>'` character. Closing tags also contain a non empty string that does not contain any `'<'` or `'>'` characters.

To be properly formatted and nested the tags in an XML file must meet the following requirements:

1. The file contains at least one set of tags and the first line of the file contains an opening tag.
2. Each opening tag must have a matching closing tag.
3. A closing tag must match the most recent opening tag. Sets of tags may not overlap.
4. There are no extraneous closing tags.

Assume that the file will have one tag or piece of data per line. Assume that all data is correctly formatted. Your method shall only check the validity of the tags. Ignore any lines that are not opening or closing tags.

Here are some examples of properly and improperly formatted XML files:

Properly formatted:
```
<DIMENSIONS>
<HT>
<FEET>
5
</FEET>
<INCHES>
9
</INCHES>
</HT>
<WEIGHT>
160
</WEIGHT>
</DIMENSIONS>
```

Improperly formatted (no tags at all):
```
12
```

Improperly formatted (closing tag does not match opening tag string):
```
<LENGTH>
12
</LN>
```

Improperly formatted (closing tag with no matching opening tag):
```
<LENGTH>
12
</LENGTH>
</DIMENSION>
```

Improperly formatted (no closing tag):
```
<LENGTH>
12
```

Improperly formatted (closing tag does not match most recent opening tag, improper nesting)
```
<DIMENSIONS>
<HT>
<FEET>
5
</FEET>
<INCHES>
9
</HT>
</INCHES>
<WT>
160
</WT>
</DIMENSIONS>
```

Write a method that returns `true` if the tags in a given XML file are properly nested, `false` otherwise. The only classes you can use are a single Stack, the Scanner class and the String methods mentioned below.

Do not use recursion on this question.

Assume you have a Stack class available with the following methods:

```
public class Stack<E> {
     public Stack() // creates a new, empty Stack
     public void push(E data)
     public E top()
     public E pop()
     public boolean isEmpty()
}
```

The String methods you may use are:

`char` **`charAt`**`(int index)` Returns the `char` value at the specified index.

`int` **`length`**`()` Returns the length of this string.

`String` **`substring`**`(int beginIndex)` Returns a new string that is a substring of this string.

`boolean` **`equals`**`(Object other)` Returns true if this String equals other.

The `Scanner` methods you may use are listed below:

`boolean` **`hasNextLine`**`()` Returns true if this scanner has another line in its input.

`String` **`nextLine`**`()` Advances this scanner past the current line and returns the input that was skipped. Recall each line will contain a single tag or piece of data.

```
// pre: fileScan connected to the start of an XML file
// post: return true if the tags in the file are properly nested,
//   false otherwise
public boolean properlyNested(Scanner fileScan) {
    Stack<String> tags = new Stack<String>();
```

5. (Using Data Structures, 15 points) This questions has two parts.

Part A (5 points). Write a method that returns the similarity score of two sets. For this question the similarity score for two sets will be the size of the intersection of the two sets times 2, minus the size of the union of the two sets

In other words, given 2 sets A and B, write a method that returns

2 * (size of $A \cap B$) - (size of $A \cup B$)

$\cap$ is the intersection operator and $\cup$ is the union operator

The method is not a part of the `Set` class.

```
// pre: set1 != null, set2 != null
// post: return 2 times the size of the intersection of set1
// and set2, minus the size of the union of set1 and set2
public <T> int getSimilarityScore(Set<T> set1, Set<T> set2)
```

Examples of return values for the `getSimilarityScore` method:

```
getSimilarityScore ( [B, C, A], [D, E]) -> returns -5
getSimilarityScore ( [B, C, A], [A, D, C, E, B]) -> returns 1
getSimilarityScore ( [], [A, B, C]) -> returns -3
getSimilarityScore ( [B, C, A], [C, B, A]) -> returns 3
```

The only methods you may use from the `Set` class are the `iterator` and `size` methods:

`Iterator<E>` **`iterator`**`()` Returns an iterator over the elements in this set.

`int` **`size`**`()` Returns the number of elements in this set.

Recall the methods from the `Iterator` interface:

`boolean` **`hasNext`**`()` Returns `true` if the iteration has more elements.

`E` **`next`**`()` Returns the next element in the iteration.

`void` **`remove`**`()` Removes from the underlying collection the last element returned by the iterator.

You may also use the `equals` method from the `Object` class.

Your method must be O(1) *space*, meaning you cannot use temporary arrays, lists, or other data structures whose size depends on the number of elements in the set.

Complete the method on the next page. The `Set` class implements the `Iterable` interface so you may use the foreach loop.

```
// pre: set1 != null, set2 != null
// post: return 2 times the size of the intersection of set1
// and set2, minus the size of the union of set1 and set2
public <T> int getSimilarityScore(Set<T> set1, Set<T> set2) {
     assert set1 != null && set2 != null;
```

Part B (10 points) A *map* is a data structure that stores key-value pairs. Each entry in the map contains a key and a value similar to words and entries in dictionaries.

Write a method that given a map of Senators, represented as Strings, and the bills Senators have voted for, represented as a Set of Integers, determines which Senators have the closest voting record, based on the `getSimilarityScore` method you implemented in Part A.

```
// pre: votingRecords != null, votingRecords.size() >= 2,
//   no values in votingRecods == null
public HashSet<String> findMostSimilarSenators(
                   Map<String, Set<Integer>> votingRecords) {
```

Return a `HashSet` of the Senators that have the largest similarity score based on the bills they have voted forThere will be at least two elements in the returned `HashSet`, but it could be larger if more than two Senators share the largest similarity score.

The methods from the Map class you may use are listed below. K is a generic type for the keys of the map and V is a generic type for the values of the map.

`Set<K>` **`keySet`**`()` Returns a `Set` of the keys contained in this map.

`V` **`get`**`(Object key)` Returns the value to which the specified key is mapped, or `null` if this map contains no mapping for the key.

The methods from the `HashSet` class you may use are:

`HashSet``()` Constructs an empty set.

`boolean` **`add`**`(E e)` Appends the specified element to this set if not already present.

`void` **`clear`**`()` Removes all of the elements from this set.

You may also use the `Set` and `Iterator` methods from part A, the `equals` method from `Object`, and the `getSimilarityScore` method from Part A. (Assume the method works regardless of what you wrote in Part A. Do not duplicate the work from Part A.)

Consider the following example. Each line represents a single entry in the map. The integers are the bill numbers the senators voted for.

```
["Ogden", (4, 1, 6, 3, 7, 5)]
["Shapiro", (5, 1, 12, 19, 13, 7, 40)]
["Smith", (5, 4, 6, 1, 3, 7)]
["Lamar", (5, 13, 19, 12, 1, 40, 7)]
["Johnson", ((5, 13, 19, 12, 1, 40, 7, 6)]
```

Given the above map, the method would return a `HashSet` with `"Lamar"` and `"Shapiro"` in it.

Complete the following method:

```
// pre: votingRecords != null, votingRecords.size() >= 2
public HashSet<String> findMostSimilarSenators (
                    Map<String, Set<Integer>> votingRecords) {
```

Name:______________________________

Answer sheet for question 1, short answer questions. **Put answers next to the letter.**

A:

B. ____

C.____

D.____

E.____

F. ____

G.____

H.____

I.____

J.____

K.____

L.____

M.____

N.____

O.____

P.____

Q.____

R.____

S.____

T.____