Points off	1	2	3	4	5		Total off	Net Score

CS 314 – Midterm 2 – Fall 2011

Your Name_____

Your UTEID _____

Circle yours TA's name: Swati Yuanzhong

Instructions:

1. There are **5** questions on this test.
2. You have 2 hours to complete the test.
3. You may not use a calculator or any other electronic devices while taking the test.
4. When writing a method assume the preconditions of the method are met.
5. When writing a method you may add helper methods if you wish.
6. When answering coding questions ensure you follow the restrictions of the question.
7. When you complete the test show the proctor your UTID. Give them the test and any scratch paper. Please leave the room quietly.

1. (2 points each, 30 points total) Short answer. Place you answers on the attached answer sheet.
   a. If a question contains a syntax error or other compile error, answer "Compile error".
   b. If a question would result in a runtime error or exception answer "Runtime error".
   c. If a question results in an infinite loop answer "Infinite loop".
   d. Recall when asked for Big O your answer should be the most restrictive correct Big O function. For example Selection Sort has an average case Big O of $O(N^2)$, but per the formal definition of Big O it is correct to say Selection Sort also has a Big O of $O(N^3)$ or $O(N^4)$. I want the most restrictive, correct Big O function. (Closest without going under.)

A. What is returned by the method call `a(5)`?

```
public int a(int x) {
     if(x == 0)
          return x;
     return a(x - 1) + x;
}
```

B. What is returned by the method call `b(0, new int[]{-4, 5, 3, 6, 9, 12, 15, 0})`?

```
public int b(int i, int[] data) {
     if(i >= data.length - 1)
          return 0;
     else if(data[i] < data[i + 1])
          return 1 + b(i + 1, data);
     else
          return b(i + 1, data);
}
```

C. What is output by the following statement? Recall the `substring` method shown returns a new `String` including all elements from the given index to the end of the `String`.

```
System.out.println(c("orange").length());

public String c(String st) {
     if(st.length() <= 1)
          return st;
     return st + c(st.substring(1)) + c(st.substring(2));
}
```

D. What is output by the following code?

```
String garbage = "1*2+y-u+(x&--v)";
Stack<Character> st = new Stack<Character>();
for(int i = 0; i < garbage.length(); i++) {
     char ch = garbage.charAt(i);
     if(Character.isLetter(ch)) {
          st.push(ch);
          st.push(ch);
     }
}

while(!st.isEmpty())
     System.out.print(st.pop() + " ");
     // pop removes and returns the top value of the stack
```

E. What does the following expression evaluate to?

```
4 1 + 3 2 + *
```

F. The `sort_F(double[] data)` method uses the selection sort algorithm. It takes the method 3 seconds to sort an array of 20,000 elements. What is the expected time for the method to sort 40,000 elements

G. Given the `sort_G(int data[])` method consider the following timing data:

- 2 seconds to sort an array with 1,000,000 distinct elements in random order.
- 8.8 seconds to sort an array with 4,000,000 distinct elements in random order
- 6 seconds to sort an array with 50,000 elements all equal to the same value
- 24 seconds to sort an array with 100,000 elements all equal to the same value

Based on the timing data, what sorting algorithm does method `sort_G` most likely use? Consider only the sorts we discussed in class.

H. If 2,000 elements are inserted into a binary search tree using the traditional, naïve algorithm what is the worst case height of the resulting tree? Give the actual number, not an order.

I. Consider the following method.

```
public List<String> create(String[] data) {
    List<String> result = new LinkedList<String>();
    for(String st : data)
        if(!result.contains(st))
            result.add(0, st); // add st to front of list
    return result;
}
```

If half of the elements in `data` are distinct and half are duplicates of other elements in `data`, what is the order (Big O) of method `create`? N = `data.length`. `LinkedList` is the Java linked list class.

J. What is the order of method `create` if the line of code

```
List<String> result = new LinkedList<String>();
```

is changed to this:

```
List<String> result = new ArrayList<String>();
```

K. You have an array with 128,000 distinct elements in unsorted order. You expect to perform 1000 searches on the array before the data changes.

Which is more efficient:

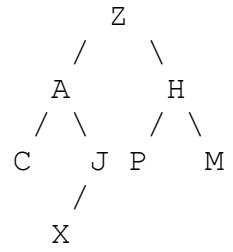
1. perform the searches without sorting
2. sort the data and them search

Justify your answer mathematically.

L. The following values are added one at a time to an initially empty binary search tree using the traditional, naïve insertion algorithm. Draw the resulting tree:

```
-15  32   15   -15  0    7
```

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



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

N. What is the result of an in-order traversal of the binary tree shown above?

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

Scratch Paper

2. Binary Trees (15 points total) Write an instance method for a `BinaryTree` class that determines the number of nodes in the tree that have 2 children. For example given the binary tree from 1.M - 1.0 the method would return 3. (Nodes that contain Z, A, and H each have 2 children.)

Recall the `BinaryTree` class:

```
public class BinaryTree<E> {

    private BTNode<E> root; // if size == 0, root == null
    private int size; // number of nodes in this tree
```

Recall the `BTNode` class:

```
public class BTNode<E> {

    public BTNode() // all values set to null
    public BTNode(BTNode<E> right, E data, BTNode<E> left)

    public E getData() // get the data stored in this node
    public BTNode<E> getLeft() // get the left child of this node
    public BTNode<E> getRight() // get the right child of this node

    // if child does not exists, null is returned

    public E setData() // set the data stored in this node
    public BTNode<E> setLeft() // set the left child for this node
    public BTNode<E> setRight() // set the right child for this node
```

(10 points) Complete the following instance method for the `BinaryTree` 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 tree.

```
/*    pre: none
      post: return the number of nodes in this tree that have 2
     child nodes.
*/
public int numNodesWithTwoChildren() {
```

Analysis for question 2. (5 points)

A. Given a `BinaryTree` that contains N nodes, what is the order (Big O) of your method?

B. Given a `BinaryTree` that contains N nodes what are the smallest and largest possible values your method will return? Do not use order / Big O to answer the question

smallest:

largest:

3. Linked Lists / Sorted Sets. (20 points) Write the `add` method for a `SortedSet` class that uses a singly linked list as its internal storage container.

The properties of the `SortedSet` class:

- The internal linked list uses singly linked nodes that store one piece of data and a reference to the next node in the list.
- The only instance variable in the `SortedSet` class is a reference to the first node in the list.
- If the list is empty the reference to the first node is set to `null`.
- The last node's next reference is set to `null`.
- The `SortedSet` and `Node` classes are generic based on Java's generic syntax.
- All elements stored in the linked list implement the `Comparable` interface. The parameter `tgt` and all elements of the list implement the `Comparable` interface. No casting is necessary.

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 sorted set.

You may not use any other classes or methods, except the `Node` class and the `compareTo` method.

Your method should be as efficient as possible given the constraints. The instance method to complete is:

```
/*   pre: val != null
     post: add val to this SortedSet unless it was already present.
     The method returns true if this SortedSet changed as a result
     of this method call, false otherwise.
*/
public boolean add(E val) {
```

Here are some example calls to the `add method` and the expected results for various `SortedSet`s of `Integer` objects.

```
[].add(12) -> returns true, set becomes [12]
[12].add(12) -> returns false, set remains [12]
[12].add(-5) -> returns true, set becomes [-5, 12]
[-5, 12].add(35) -> returns true, set becomes [-5, 12, 35]
[-5, 12, 35].add(17) -> returns true, set becomes [-5, 12, 17, 35]
[-5, 12, 17, 35].add(35) -> returns false, set remains [-5, 12, 17, 35]
```

Recall the `Node` class:

```
public class Node<E extends Comparable<E>> {
     public Node(E data, Node<E> next)

     public E getData()
     public Node<E> getNext()

     public void setData(E data)
     public void setNext(Node<E> n)
```

```
public class SortedSet<E extends Comparable<E>> {

     private Node<E> first; // first == null if empty

// Complete the following instance method for the SortedSet class.

/*   pre: val != null
     post: add val to this SortedSet unless it was already present.
     The method returns true if this SortedSet changed as a result
     of this method call, false otherwise.
*/
public boolean add(E val) {
```

4. Sets, Iterators (17 points total) A *Fuzzy Set* is a set in which membership in the set is expressed as a *degree* which varies between 0 and 1 inclusive. In a traditional set, a given value is either in the set or it isn't. In other words, for traditional sets, the degree is either 0 (not in the set) or 1 (in the set). For fuzzy sets the degree of a value will vary between 0 (not in set) to 1 (absolutely in set), but may be a fractional value between 0 and 1.

Fuzzy sets are used in areas such as bioinformatics and genealogical research.

Complete a method in the `FuzzySet` class that creates and returns the *fuzzy intersection* of two `FuzzySet`s.

The fuzzy intersection of two fuzzy sets consists of all the elements the sets have in common. The degree of an element in the new fuzzy set is equal to the product of the degrees of the elements from the two original sets. Consider the this example: (The elements are shown in order for clarity. The elements are not guaranteed to be sorted.)

```
First Set:  [(A, 1.0), (C, 0.1), (D, 0.5), (G, 0.7), (H, 0.2)]
Second Set: [(A, 0.5), (B, 0.5), (C, 0.7), (E, 0.1), (F, 1.0), (G, 0.02)]
```

The fuzzy intersection of those two sets is: `[(A, 0.5), (C, 0.07), (G, 0.014)]`

Write your method in terms of other methods in the `FuzzySet` class. do not assume any instance variables for the class.

You may use these methods from the `FuzzySet` class

- `FuzzySet()` Create a new, empty `FuzzySet`.
- `boolean` **`add`**`(SetPair val)` Add `val` to this set. If the element in `val` was already present replace the old degree with the degree in `val`. Returns true if this set was changed as a result of this method call. (Returns true if the element in `val` was not present or if the it was, but the degree has been changed.)
- `Iterator<E>` **`iterator`**`()` Returns an iterator over the elements in this set.

Recall these 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

The `FuzzySet` class stores `SetPair` objects:

```
public class SetPair<E> {

     // pre: elem != null, 0 <= degree <= 1.0
     public SetPair(E elem, double degree)

     public E getElem()
     public double getDegree()
```

```
    // more instance methods for SetPair
    public void setElem(E newElem) // pre: newElemn != null
    public void setDegree(double degree) // pre: 0 <= degree <= 1.0
```

You are not allowed to use any methods except those listed in this question and the `equals` method. The `FuzzySet` class is not `Iterable` so you may not use the for-each loop.

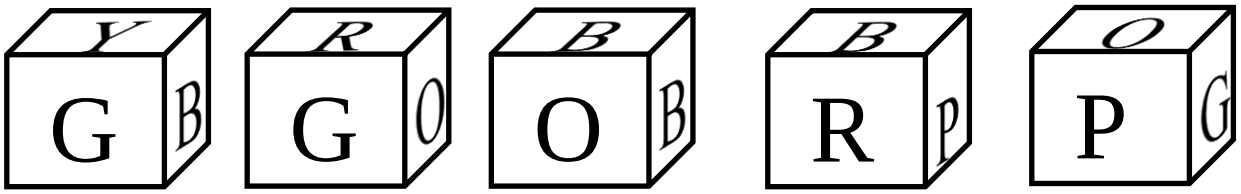
Complete the following instance method for the `FuzzySet` class.

```
public class FuzzySet<E> {

    // do not assume or create any instance variables

/*  pre: other != null
    post: return a FuzzySet that is the fuzzy intersection of this
    FuzzySet and other. Neither this or other are altered as a result
    of this method call.
*/
public FuzzySet<E> getFuzzyIntersetion(FuzzySet<E> other) {
```

5. Recursion. (18 points) Write a method to find a solution to a dice puzzle.

- The puzzle consists of a variable number of dice.
- Each face on the dice is a color, not a number.
- **The order of the dice is fixed.** None of the dice may be moved to a different position in the line.
- Each die may be rotated to alter the position of the faces of the die.

Consider the example with five dice. Letters are used to designate colors. Note, the actual number of dice will vary.



Die 1 Die 2 Die 3 Die 4 Die 5

The goal of this puzzle is to line up the dice so when two faces from separate dice touch, they are the same color. In other words find a solution to the puzzle so the face of die 1 and the face of die 2 that touch are the same color, the face of die 2 and the face of die 3 that touch are the same color, and so forth. Each pair of touching faces must be the same color, but the sets of faces do not have to be the same color. For example the touching faces of die 1 and die 2 could be blue, but the touching faces of die 2 and die 3 could be orange.

Complete the following method:

```
/*   pre: dice != null, dice.size > 0
     post: return an ArrayList of Die objects positioned so that the
     puzzle is solved. The order of the dice in the ArrayList must be
     the same as in the array dice, unless there is no solution. If
     there is no solution, return an empty ArrayList.
*/
public ArrayList<Die> solvePuzzle(Die[] dice)
```

Recall these methods from the `ArrayList` class:

```
public ArrayList() // construct an empty ArrayList
public boolean add(E obj) // add obj to the end of the ArrayList
public int size() // return the number of elements in the ArrayList
public E remove(int pos) // remove and return the element at index pos
public E get(int pos) // return the element at the given position
```

The methods for the `Die` class are on the next page. The `Die` class models a six sided die. Each face of the die has a color represented by a `char`. There is a method to set the position of the die by specifying which face of the `Die` is facing left. The faces of the `Die` are numbered 0 to 5.

```
public class Die {
     // constructors not shown

     // pre: 0 <= side <= 5
     // return the color on the given side
     public char getColor(int side)

     //pre: 0 <= side < = 5
     // Convenience method to get the color on the side opposite the
     // given side. Opposite sides are paired as follows:
     // 0 and 3, 1 and 4, 2 and 5
     public char getColorOppositeSide(int side)

     // pre: 0 <= side <= 5
     // set the position of this dice so the given side is facing
     // left. The opposite side is facing right.
     public void positionLeftFace(int side)

     // pre: none
     // return the side that is currently facing left. The side
     // opposite the returned side is facing right.
     public int getLeftFacingSide()

}
```

You are not allowed to use any methods except those listed in this question.

Complete the `solvePuzzle` method on the next page. (Hint: Create a recursive helper method.)

```
/*   pre: dice != null, dice.size > 0
     post: return an ArrayList of Die objects positioned so that the
     puzzle is solved. The order of the dice in the ArrayList must be
     the same as in the array dice, unless there is no solution. If
     there is no solution, return an empty ArrayList.
*/
public ArrayList<Die> solvePuzzle(Die[] dice)
```

COMPLETE ON THE NEXT PAGE

COMPLETE ON THE NEXT PAGE

```
/*   pre: dice != null, dice.size > 0
     post: return an ArrayList of Die objects positioned so that the
     puzzle is solved. The order of the dice in the ArrayList must be
     the same as in the array dice, unless there is no solution. If
     there is no solution, return an empty ArrayList.
*/
public ArrayList<Die> solvePuzzle(Die[] dice) {
```

Question 1 answer Sheet.

Name_____

A. _____

B. _____

C. _____

D. _____

E. _____

F. _____

G. _____

H. _____

I. _____

J. _____

K. _____

L. _____

M. _____

N. _____

O. _____