Points off	1	2	3	4	5		Total off	Net Score

CS 314 – Exam 2 – Spring 2015

Your Name_____

Your UTEID _____

Instructions:

1. There are **5** questions on this test. 100 points available. Scores will be scaled to 200 points.
2. You have 2 hours to complete the test.
3. Place you answers on this test. Not the scratch paper.
4. You may not use a calculator or any other electronic devices while taking the test.
5. On coding questions you may add helper methods.
6. You methods shall be as efficient as possible in terms of time and space given the restrictions of the question.
7. When answering coding questions, ensure you follow the restrictions of the question.
8. Test proctors will not answer any questions regarding the content of the exam. If you think a question is ambiguous or has an error, state your assumptions and answer based on those assumptions.
9. When you complete the test show the proctor your UTID, give them the test and all the scratch paper, used or not, and leave the room quietly.

1. (1 point each, 20 points total) Short answer. Place your answer on the line next to or under the question. Assume all necessary imports have been made.

   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(-7)`? _____

```
public int a(int x) {
      if(x >= 2)
            return 5;
      else
            return 2 + a(x + Math.abs(x) / 2 + 1);
}
```

B. What is returned by the method call `b("ALAN_TURING")` _____

```
public String b(String s) {
     if(s.length() <= 1)
          return "!";
     else
          return s.charAt(s.length() - 1)
                    + b(s.substring(0, s.length() - 2));
}
```

C. What is returned by the method call `c(4)` _____

```
public int c(int x) {
     if(x <= -1)
          return 10;
     else
          return 5 + c(x - 2) + c(x - 1);
}
```

D. Consider the following code that uses the classes from Assignment 9, the set assignment.

What is output by the following code? _____

```
SortedSet<Integer> s1 = new AbstractSet<Integer>();
s1.add(7);
s1.add(5);
s1.add(7);
System.out.print(s1.toString());
```

E. What is the worst case order (Big O) of the following method? _____

```
public int[] e(LinkedList<Integer> list) {
     int[] result = new int[list.size()];
     Random r = new Random();
     for(int i = 0; i < result.length; i++) {
          int index = r.nextInt(list.size());
          if(list.get(index) < index)
               result[i] = list.get(index);
     }
     return result;
}
```

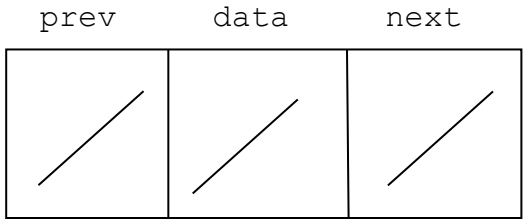
F. What is the worst case order (Big O) of the following method if the parameter is a Java `LinkedList`?

_____

```
public void f(List<Integer> list, int tgt) {
     Iterator<Integer> it = list.iterator();
     while(it.hasNext()) {
          if(it.next() >= tgt)
               it.remove();
     }
}
```

G. What is the worst case order of method `f` from question f if the parameter is a Java `ArrayList`?

H. Draw the variables, references, and objects that exist after the following code executes. Draw node objects as shown below and boxes for variables. The example has all instance variables set to `null`. The example does not show any of the variables that actually refer to the node object. You must show all variables and their references in your drawing. Use arrows to show references and a forward slash to indicate variables that store `null`. Assume the node class is the doubly linked node from the linked list assignment and that the fields of the class are all `public`.




```
DoubleListNode<Object> n1
     = new DoubleListNode <Object>(null, null, null);
                  // parameters are prev, data, next

DoubleListNode<Object> n2 =
     = new DoubleListNode <Object>(n1, n1, null);

n2.next = n2;
n1.next = n2.next.prev;
n2.data = n1.next.prev;
DoubleListNode<Object> n3 = n2.prev.next;
```

I. A method uses the binary search algorithm on an array of ints. It takes 10 seconds for the method to complete 10,000 searches on an array with 1,000,000 elements. What is the expected time to complete 50,000 searches on an array with 2,000,000 elements. The arrays are both already sorted.

_____

J. Consider the following timing data for a method that sorts arrays of doubles

Number of elements	Time to sort array with elements in random order.	Time to sort array with elements already in ascending order.
10,000	1 second	.01 seconds
20,000	4 seconds	.02 seconds
40,000	16 seconds	.04 seconds

Which sorting algorithm we studied does the method most likely use?

_____

K. A method uses the selection sort algorithm to sort data. The method takes 20 seconds to sort an array with 100,000 distinct elements in random order. What is the expected time for the method to sort an array with 400,000 distinct elements in random order?

_____

L. What is the result of the following postfix expression? (single integer for answer) _____

```
5 6 - 12 3 + * 2 -
```

M. Write an infix expression that is equivalent to the following postfix expression.

_____

```
3 2 + 17 4 - *
```

N. What is output by the following code? Assume the `Queue314` class is a traditional queue class like the one we implemented in lecture.

_____

```
Queue314<Integer> q = new Queue314<Integer>();
for(int i = 5; i > 1; i -= 2)  {
     q.enqueue(i);
     q.enqueue(q.front() + i);
     q.enqueue(q.front());
}

while(!queue.isEmpty())
     System.out.print(q.dequeue() + " ");
```

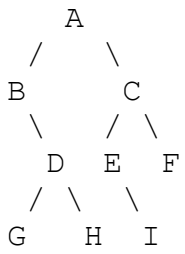
O. What is output by the following code? The code uses the Java `Stack` class.

_____

```
Stack<Character> st = new Stack<Character>();
String data = "EWDijkstra";
for(int i = 0; i < data.length(); i += 2)
     st.push(data.charAt(i));

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

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



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

_____

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

_____

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

_____

S. The following values are inserted one at a time into a binary search tree using the simple, naïve algorithm demonstrated in class. Draw the result tree.

```
10  84  6  20  22  33
```

T. The binary search tree class in the code below uses the simple, naïve algorithm demonstrated in class. It takes 1 seconds for the method to complete when the array has 1,000 distinct elements in random order. What is the expected time for the method to complete when the array has 1,000,000 distinct elements in random order?

_____

```
public BST<Integer> makeTree(int[] data) {
     BST<Integer> result = new BST<Integer>();
     for(int x : data)
          result.add(x);
     return result;
}
```

2. stacks - 20 points. Write a method that removes any consecutive duplicate items from a Queue. Use a single `Stack` as an auxiliary storage container.

Assume the `Stack` class in this question has the `push`, `pop`, `top`, and `isEmpty` methods and a zero argument constructor that creates an empty `Stack`. Assume the `Queue` class has the `enqueue, dequeue, front,` and `isEmpty` methods.

Examples:

Initial `Queue`		Resulting `Queue`	
front	back	front	back
`[5, 3, 1, 0, 5, 2, 1]`		`[1, 2, 5, 0, 1, 3, 5]`	

This queue has no consecutive duplicate items so the resulting queue will contain those same items, although the order will be reversed

Initial `Queue`		Resulting `Queue`	
front	back	front	back
`[5, 5, 5, 0, 0, 1, 0, 0]`		`[0, 1, 0, 5]`	

This queue has 3 consecutive 5's and two sets of 2 consecutive zeros. In the resulting queue 2 of the 5's will no longer be present and 1 of each of the 0's from the 2 runs will no longer be present.

You may use one `Stack` as an auxiliary storage container and the `equals` method. You may not use any other Java class or methods,

```
// pre: q != null
public <E> void removeConsecutiveDuplicates(Queue<E> q) {
```

3. linked lists - 20 points. Complete the `isSorted` instance method for the `LinkedList314` class. The method is an accessor. The method returns true if the elements of the linked list are in sorted ascending order based on the `compareTo` method.

- You may not use any other methods in the `LinkedList314` class unless you implement them yourself as a part of your solution.
- The `LinkedList314` class uses singly linked nodes.
- The list has references to the first node in the chained structure of nodes.
- When the list is empty, `first` is set to `null`.
- None of the data in the list equals `null`.
- If the list is not empty the last node in the list has its next reference set to `null`.
- You may use the `Node` class and the `Comparable compareTo` method.
- **You may not use any other Java classes or native arrays.**

```
public class LinkedList314<E extends Comparable<? super E>> {

     private Node<E> first; // refers to first node in the chain of nodes
}
```

The `Node` class.

```
public class Node<E> {
     public Node(E item, Node<E> next)
     public E getData()
     public Node<E> getNext(
     public void setData(E item)
     public void setNext(Node<E> next)
}
```

Examples.

```
[].isSorted() ->  returns true

[1].isSorted()  ->  returns true

[1, 1].isSorted()  ->  returns true

[1, 2, 1].isSorted()  ->  returns false

[1, 2, -3].isSorted()  ->  returns false

[-3, 5, 7, 11, 11].isSorted()  ->  returns true
```

Complete the method on the next page.

Complete the following method instance method of the `LinkedList314` class.

```
/* pre: none
   post: return true if the elements in this linked list are sorted in
     ascending order based on the natural ordering of the elements. */

public boolean isSorted() {
```

4. maps - 20 points. Write a method that determines which problems from a practice problem website have been solved the most by the students in a class.

A map stores the names of students (a `String`) as keys. The values are `Set`s of `Integer`s that represent the problems solved by the associated student / key. For example:

```
"David", (5, 1, 6, 10, 12)
"Olivia", (6, 12, 37, 42, 1, 10, 11, 20)
"Kelly", (11, 17, 5, 37, 42, 21, 56)
"Isabelle", (13, 15, 17, 2, 18, 19, 5, 1)
"Mike", (6)
```

The method returns a set of the problems solved by the most students. In the example above problems 1, 5, and 6 were all solved by three students. These integers would be returned in a set.

Recall the following methods from the `Map` interface.

- `Set<K>` **`keySet`**`()` - Returns a Set view 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.
- `V` **`put`**`(K key, V value)` - Associates the specified value with the specified key in this map
- `V` **`remove`**`(K key)` - Removes the mapping for a key from this map if it is present.
- `boolean` **`containsKey`**`(K key)` - Returns true if key is in this mapping.

You may obtain iterators for `Collection`s and use the methods from the `Iterator` interface.
Recall `Map`s do not have an `Iterator iterator()` method.

You may use a `HashMap<Integer, Integer>` as an auxiliary data structure and a Java `TreeSet` for the result. You only need the `TreeSet` constructor and the `add` method.
You may not use any other auxiliary data structures.

```
//   pre: solved != null, none of the values in solved == null
//        solved.size() > 0, each value in solved size() > 0
//   post: per the question description
//   The parameter solved is not altered as a result of this method.

public TreeSet<Integer> getMostSolvedProblems(
                              Map<String, Set<Integer>> solved) {
```

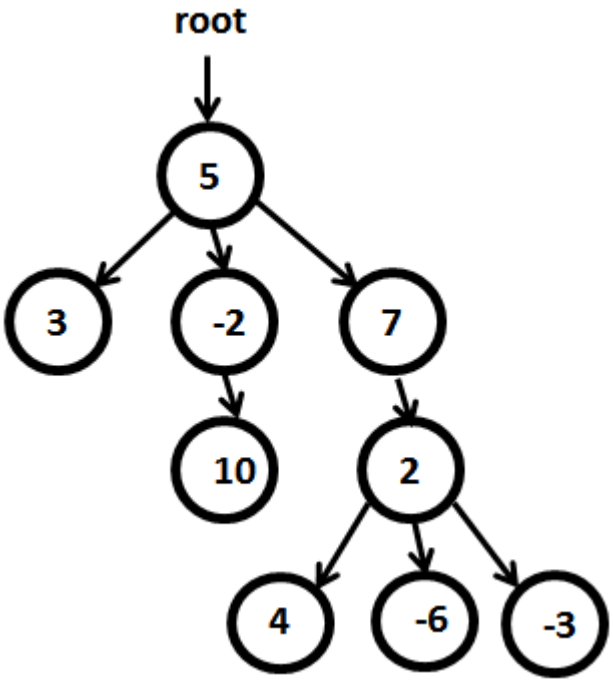
Complete this method on the next page:

```
//   pre: solved != null, none of the values in solved == null
//   post: per the question description
//   The parameter solved is not altered as a result of this method.

public TreeSet<Integer> getMostSolvedProblems(
                                Map<String, Set<Integer>> solved) {
```

5. recursion and trees. 20 points. In this question you have a tree that stored ints in each node. The tree is NOT a binary tree. Nodes may have any number of children. Determine if there is a path in the tree from the root to descendant nodes so that the sum of the values in the nodes in the path equal some target value.

The path must start at the root and descend down the tree following the links that exist in the tree. The value of a node in the path must be included.



Given the tree to the left, the `hasPath` method would return the following:

```
hasPath(5) -> true: 5 in root

hasPath(3) -> true: 5, -2

hasPath(13) -> true: 5, -2, 10

hasPath(18) -> true: 5, 7, 2, 4

hasPath(0) -> true (path with 0 nodess)

hasPath(20) -> false

hasPath(1) -> false

hasPath(7) -> false (must use 5 in root)
```

Assume the children of a node are numbered 0 to N - 1, left to right.

Complete the instance method `hasPath(int tgt)` for the `IntTree` class.

```
public class IntTree {
     // if tree empty, root = null
     private IntNode root;

     private static class IntNode {

          // value stored in this node
          private int data;

          // list of child nodes. If leaf node, children.size() == 0
          private ArrayList<IntNode> children;

          // rest of class not shown
     }
```

Note, if the tree were empty `hasPath(0)` returns `true`. If the tree were empty, `hasPath` of any non-zero `int` returns `false`.

Complete the following instance method of the `IntTree` class. You may not use any classes other than the methods of the Java `ArrayList`. Do not use any auxiliary data structures.
Add a recursive helper method.

```
// pre: none
// this IntTree is not altered as a result of this method call
// post: return true if there is a path starting at the root of this
// tree and proceeding down the tree so that the sum of the values in
// in the path equal tgt.
public boolean hasPath(int tgt) {
```