Points off	1	2	3	4	5		Total off

CS 314 – Exam 2 – Fall 2017

Your Name_____________________________________

Your UTEID ___________________________________

Circle your TAs Name: Gilbert Jacob Jorge Joseph Lucas Rebecca Shelby

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 your final answers on this test. Not on scratch paper. Answer in pencil.
4. You may not use electronic devices of any kind while taking the test.
5. When answering coding questions, ensure you follow the restrictions of the question.
6. Do not write code to check the preconditions.
7. On coding questions, you may implement your own helper methods.
8. On coding questions make your solutions as efficient as possible given the restrictions of the question.
9. 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.
10. 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 only results in a runtime error or exception, answer **runtime error**.
   c. If a question only 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(2)`? _________________________

```
public static int a(int x) {
     if (x > 20)
          return -3;
     else
          return a(x * 3) + x / 3;
}
```

B. What is output by the method call `b(7)` ? ____

```
public static int b(int x) {
     if (x < 0) {
          System.out.print("!" + x);
          return x * -2;
     } else {
          System.out.print(x);
          x = b(x - 3) + 2;
          System.out.print(x);
          return x;
     }
}
```

C. What is returned by the method call `c(20, 5)`? _________________________

```
public static int c(int x, int y) {
     if (x <= y)
          return 1;
     else
          return y + c(x - y, y + 2);
}
```

D. What is output by the following code? _________________________

```
System.out.print(d("01234567").length());

public static String d(String s) {
     if (s.length() < 3)
          return s + s;
     else
          return "*" + d(s.substring(1)) + d(s.substring(3));
}
```

E. The method call `e(30)` takes 1 second to complete.
What is the expected time for the method call `e(40)` to complete?

____

```
public static int e(int x) {
     if (x == 0)
          return 2;
     else
          return e(x - 1) + x + e(x - 1);
}
```

F. The following method takes 3 seconds to complete when `list` is a Java `LinkedList` with a size equal to 100,000 and half of the elements in `list` are greater than `target`. What is the expected time for the method to complete when `list` is a Java `LinkedList` with a size equal to 400,000 and half of the elements in `list` are greater than `target`?

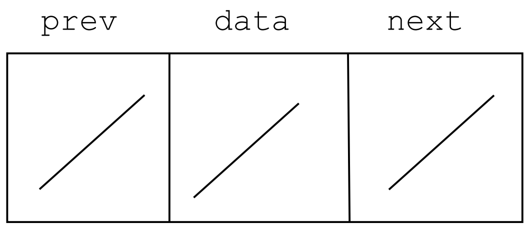
____

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

G. The method from part F takes 5 seconds to complete when `list` is a Java `ArrayList` with a size equal to 10,000 and all of the elements in `list` are greater than `target`. What is the expected time for the method to complete when `list` is a Java `ArrayList` with a size equal to 100,000 and all of the elements in `list` are greater than `target`?

____

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<Integer> n1
     = new DoubleListNode<>(null, new Integer(7), null);

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

n1.prev = n2;
n1.prev.next.next = n1;
n2.data = n1.prev.next;
n1 = n2;
```

I. What is the worst case order (Big O) of the following method? N = `list.size()`. `list` is a Java `LinkedList` __________________

```
public static double im(LinkedList<Double> list) {
     double result = 0;
     for (int i = 0; i < list.size() / 2; i++) {
          for (int j = list.size() / 2; j < list.size(); j++) {
               if (list.get(i) < list.get(j)) {
                    result += list.get(j) * list.get(i);
               }
          }
     }
     return result;
}
```

J. What does the following postfix expression evaluate to? Single integer for answer.

```
100 10 / 10 5 * 2 4 - * -
```

____

K. The `StackExam2` class implements a stack and uses a singly linked list like the one presented in lecture as its internal storage container. The end of the list represents the top of the stack. The following method takes 2 seconds to complete when there are 10, 000 elements in the stack. What is the expected time for the method to complete when there are 40,000 elements in the stack?

____

```
public static int k(StackExam2 <Integer> st) {
     int total = 0;
     StackExam2 <Integer> t = new StackExam2<>();
     while (!st.isEmpty()) {
          total += st.top();
          t.push(st.pop());
     }
     while (!t.isEmpty())
          st.push(t.pop());
     return total;
}
```

L. It takes a method using the mergesort algorithm 10 seconds to complete when sorting an array with 1,000,000 items in random order. What is the expected time for the method to complete when sorting an array with 4,000,000 items in random order?

____

M. It takes a method using the insertion sort algorithm 5 seconds to complete when sorting an array with 50,000 items in random order. What is the expected time for the method to complete when sorting an array with 100,000 items in random order?

____

N. What is output by the following code? The `Queue314` class is the same as the one presented in lecture.

____

```
Queue314<Integer> q = new Queue314<>();
for (int i = 2; i < 9; i++) {
     if (i % 2 == 0)
          q.enqueue(i);
     else {
          q.enqueue(q.front());
          q.dequeue();
          q.enqueue(q.front());
     }
}
while (!q.isEmpty()) {
     System.out.print(q.front() + " ");
     q.dequeue();
}
```

O. What is output by the following code? ______________________________

```
ArrayList<Integer> a1 = new ArrayList<>();
a1.add(12);
ArrayList<Rectangle> a2 = new ArrayList<>();
a2.add(new Rectangle(0, 0, 12, 12)); // x, y, width, height
methodO(a1, a1);
methodO(a1, a2);


public static void methodO(ArrayList a1, ArrayList a2) {
     Comparable c1 = (Comparable) a1.get(0);
     Comparable c2 = (Comparable) a2.get(0);
     System.out.print(c1.compareTo(c2));
}
```

P. In a complete binary tree with 10 nodes, how many nodes are there with 2 children? _______________

Q. The following values are added one at a time to an initially empty binary search tree using the simple algorithm demonstrated in class. What is the results of a post order traversal of the resulting tree?

```
17  -5  0  12  25  -5  37  -5  26  20
```

____

R. What is the height of the root node in the resulting tree from part Q? _______________

S. The following method takes .001 seconds to complete when n = 100,000. What is the expected time for the method to complete when n = 200,000. The `BST` class is the same as the one presented in lecture. It using the simple add algorithm, but implements it iteratively instead of recursively. The `nextInt` method is O(1).

____

```
public static BST<Integer> s(int n) {
     BST<Integer> result = new BST<>();
     Random r = new Random();
     for (int i = 0; i < n; i++) {
          int t = r.nextInt() % 5;
          result.add(t);
     }
     return result;
}
```

T. The following method takes 4 seconds to complete when `data.length = 10,000`. What is the expected time for the method to complete when `data.length = 20,000`. In each case, the elements in `data` are distinct (no repeats) and are initially in random order. The `BST` class is the same as the one presented in lecture. It using the simple add algorithm, but implements it iteratively instead of recursively.

____

```
public static BST<Integer> t(int[] data) {
     BST<Integer> result = new BST<>();
     Arrays.sort(data); // assume mergesort
     for (int x : data) {
          result.add(x);
     }
     return result;
}
```

2. Linked Lists I (20 points) - Complete the `rangeIsEqual` instance method for the `LinkedList314` class. The method determines if two linked lists have the same elements, in the same order for a given range of indices.

- **<u>You may not use any other methods in the `LinkedList314` class unless you implement them yourself as a part of your solution.</u>**
- The `LinkedList314` class uses singly linked nodes.
- The list has a reference to the first node in the chain of nodes and a size variable.
- When the list is empty, `first` is set to `null`.
- The lists do not store null values.
- If the list is not empty, the last node in the chain of nodes, has its next reference set to `null`.
- You may use the nested `Node` class and the `equals` method for `Object`s.
- **You may not use any other Java classes or native arrays.**

```
public class LinkedList314<E> {

     private Node<E> first;  // refers to first node in chain of nodes.
     private int size; // number of nodes and elements in this list

     private static class Node<E> { // The nested Node class.
          private E data;
          private Node<E> next;
     }
}
```

Examples of calls to `rangeIsEqual(LinkedList314<E> other, int start, int stop)`. The range is specified by start inclusive to stop exclusive.
In these examples the lists contain `Integer` objects.

```
[5, 12, 7].rangeIsEqual([3, 12, 7, 37], 0, 2) -> false (5 != 3)

[5, 12, 7].rangeIsEqual([3, 12, 7, 37], 1, 2) -> true

[5, 12, 7].rangeIsEqual([3, 12, 7, 37], 1, 3) -> true

[5, 12, 7].rangeIsEqual([3, 12, 7, 37], 1, 4) -> false, index 3 not
valid for calling list

[5, 12, 7].rangeIsEqual([], 0, 1) -> false

[5, 5, 12, 5, 5].rangeIsEqual([4, 5, 12, 5], 1, 4) -> true

[5, 5, 12, 5, 5].rangeIsEqual([4, 5, 12, 5], 1, 5) -> false, index 4
not valid for explicit argument list

[5, 5, 12, 5, 5].rangeIsEqual([4, 5, 12, 5], 3, 4) -> true
```

Complete the following `rangeIsEqual` method for the `LinkedList314` class.

```
/* pre: other != null, 0 <= start, 0 < stop, start < stop
   post: per the question description. Neither list is altered. */
public boolean rangeIsEqual(LinkedList314<E> other, int start, int stop) {
```

3. Linked Lists II (20 points) - Complete the `removeRange` instance method for the `LinkedList314` class. The method removes all the specified elements in the given range.

This question uses the **<u>same</u>** `LinkedList314` class as question 2.

- **<u>You may not use any other methods in the `LinkedList314` class unless you implement them yourself as a part of your solution.</u>**
- **You may not use any other Java classes or native arrays.**

```
public class LinkedList314<E> {

     private Node<E> first;  // Refers to first node in chain of nodes.
     private int size; // number of nodes and elements in this list

     // The nested Node class.
     private static class Node<E> {
          private E data;
          private Node<E> next;
     }
}
```

Examples of calls to `removeRange(int start, int stop)`.
The range is specified by start inclusive to stop exclusive.
In these examples the lists contain `Integer` objects.

```
[5, 12, 7, 31, 42].removeRange(3, 3) -> [5, 12, 7, 31, 42]

[5, 12, 7, 31, 42].removeRange(5, 5) -> [5, 12, 7, 31, 42]

[5, 12, 7, 31, 42].removeRange(1, 3) -> [5, 31, 42]

[5, 12, 7, 31, 42].removeRange(0, 1) -> [12, 7, 31, 42]

[5, 12, 7, 31, 42].removeRange(0, 3) -> [31, 42]

[5, 12, 7, 31, 42].removeRange(4, 5) -> [5, 12, 7, 31]

[5, 12, 7, 31, 42].removeRange(0, 5) -> []
```

Complete the method on the next page.

```
/* pre: 0 <= start <= size, 0 <= stop <= size, start <= stop
   post: Elements from start inclusive to stop exclusive removed,
     size updated correctly. */
public void removeRange(int start, int stop) {
```

4. Trees (20 points) - Implement an instance method for a binary tree class that determines the number of parent - child pairs in the tree whose stored values, when added together, equal a given target value.

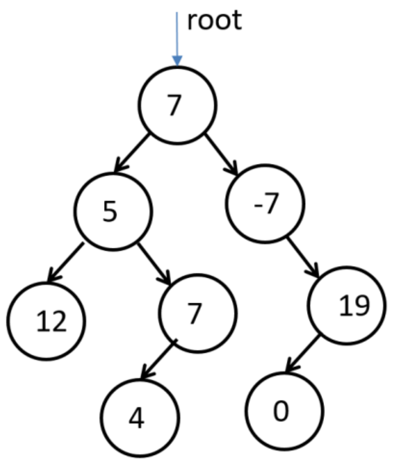
```
public class BinaryTree {

     private BNode root; // root == null if tree is empty

     // nested BNode class
     private static class BNode {

          private int data;
          // left and right child are null if no child
          private BNode left;
          private BNode right;
     }
}
```

Consider the following tree, `t`. This is not a binary search tree.



```
t.parentChildPairsWhoseSumEquals(12) returns 3

t.parentChildPairsWhoseSumEquals(0) returns 1

t.parentChildPairsWhoseSumEquals(19) returns 1

t.parentChildPairsWhoseSumEquals(5) returns 0

t.parentChildPairsWhoseSumEquals(20) returns 0

t.parentChildPairsWhoseSumEquals(-12) returns 0
```

Do not create any new nodes or other data structures. You may use the `BNode` class, but do not use any other methods or classes. You may add your own helper method.

```
/*   pre: none
     post: Per the problem description.
     This BinaryTree is not altered as a result of this method call.*/
public int parentChildPairsWhoseSumEquals(int tgt) {
```

5. Recursive Backtracking (20 points) Write a method that determines if it possible to fill a given number of shipping trucks to their weight capacity given a list of very heavy, very small items to ship.

Our goal is to fill each truck to its weight capacity. The total weight of the items to ship will always equal or exceed the total capacity of our trucks. In many cases we will have left over items which is acceptable. The goal is not to ship all the items, rather to see if we can maximize the shipping capacities of the trucks. All of the items would fit into a single truck, but we are limited by the weight capacities of the trucks.

Each truck has a given weight capacity expressed as an integer. Each item to be shipped has a weight, also an integer.

Our method will return true if we can fill each truck to its weight capacity via some combination of items, false otherwise.

Consider the following examples:

1. We have 3 trucks with weight capacities of 3, 3, and 4.
The items available have weights of 1, 2, 2, 2, 2, 2, 4.

There is no way to fill each truck to capacity. We could put a 2 and 1 into the first truck with a weight capacity of 3. We could put either the 4 or two 2's into truck number 3 with a weight capacity of 4. But truck 2 cannot be filled to its weight capacity. We only have items with a weight of 2 left.

2. We have 4 trucks with weight capacities of 3, 4, 4, and 6.
The items available have weights of 1, 2, 2, 2, 2, 2, 2, 4, 4.
One valid set up is [(1, 2), (2, 2), (4), (4, 2)].
Each set of parenthesis represents the items in a truck.

Note, the examples show the truck capacities and items weight in sorted order, but do not assume this is true.

Complete a `canShipHelp` helper method that uses recursive backtracking and returns a `true` if there is some combination of the given items or a subset of the given items so we can fill each truck to its weight capacity, `false` otherwise.

Do not use any other Java class or methods in your answer except the array `length` field.

Here is the initial method

```
/* pre: trucks != null, trucks.length > 0, all elements of trucks > 0,
   items != null, all elements of items > 0,
   the sum of the elements in items >= the sum of the elements in
   trucks.
   post: neither trucks or items is altered by the completion of this
   method. Return a value as described in the question. */
public boolean canShip(int[] trucks, int[] items) {
```

Complete the arguments for `canShipHelp` and the `canShipHelp` method on the next page.

Do not add any code to the original `canShip` method.

```
public boolean canShip(int[] trucks, int[] items) {

     return canShipHelp(
}
```