Points off	1	2	3	4	5		Total off	Net Score

CS 314 – Exam 2 – Fall 2019

Your Name: _____

Your UTEID: _____

Circle your TAs Name: **Amir Andrew Ben Claire Ethan Hailey Jacob Nina Sam Suhas Smruti Terrell Tony**

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 the scratch paper. **Answer in pencil**.
4. You may not receive any outside assistance. No electronic devices or calculators may be used.
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 unless disallowed by the question.
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 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. Assume $\log_2(1{,}000) = 10$ and $\log_2(1{,}000{,}000) = 20$.

A. What is output by the method call `a(15)`? _____

```
// method used for 1.a and 1.b
public static void a(int n) {
     System.out.print("1");
     if (n >= 10)
          a(n / 10);
     if (n % 2 == 0)
          System.out.print("2");
     else
          System.out.print("3");
}
```

B. What is output by the method call `a(3294)`? _____

C. What is returned by the method call `c(1)`? _____

```
// method used for 1.c and 1.d
public static int c(int n) {
     if (n >= 4)
          return 3;
     else
          return c(n + 1) + 2 + c(n + 1);
}
```

D. The method call `c(-20)` takes 0.01 seconds to complete. _____
What is the expected time for the method call `c(-25)` to complete?

E. What is output by the following code? _____

```
int[] dataE = {6, 4, 9, 3, -6};
int[] count = {0};
System.out.print(e(dataE, count, 0, 4));

// method used for 1.e and 1.f
public static int e(int[] d, int[] ct, int o, int h) {
     ct[0]++;
     if (o == h)
          return d[o];
     else if (o > h)
          return 0;
     else {
          int m = (o + h) / 2;
          return d[m] + e(d, ct, m + 1, h) + e(d, ct, o, m - 1);
     }
}
```

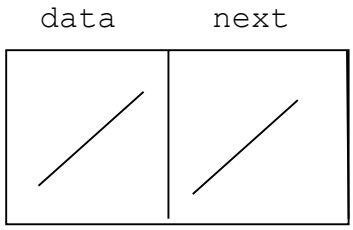
F. What is output by the following code? _____

```
int[] dataF = {6, 4, 9, 3, -6};
int[] count = {0};
e(dataF, count, 0, 4);
System.out.print(count[0]);
```

G. This question refers to the Java `List` interface and the Java **`AbstractList`** class which is abstract and implements the Java **`List`** interface. What is output by the following code? _____

```
AbstractList<String> listH = new AbstractList<>();
System.out.print(listH.size() + " ");
listH.add("CS314");
System.out.print(listH.size() + " ");
```

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 all 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 singly linked node from the linked list examples we did in class and that the fields of the class are all `public`.




```
// params are (data, next)
Node<Object> n1 = new Node<>(null, new Node<Object>("CH", null));
Node<Object> n2 = new Node<>(n1.data, n1.next);
n1.data = n2;
n1 = n2;
```

Answer:

I. The following method takes 1 second to complete when the list contains 20,000 elements.
What is the expected time for the method to complete when the list contains 80,000 elements?
None of the elements in the list equal `null`. `LinkedList` is the Java `java.util.LinkedList` class.

_____

```
public static int i(LinkedList<String> list) {
     int result = 0;
     while (list.size() != 0) {
          result += list.remove(0).length();
     }
     return result;
}
```

J. The following method takes 5 seconds to complete when the list contains 10,000 elements. _____

What is the expected time for the method to complete when the list contains 30,000 elements?
None of the elements in the list equal `null`. In each case roughly one quarter of the elements in the list have a length equal to `tgt`.
The elements with a length equal to `tgt` are spread evenly throughout the list.
`LinkedList` is the Java `java.util.LinkedList` class.

```
public static int j(LinkedList<String> list, int tgt) {
     int result = 0;
     for (int i = list.size() - 1; i >= 0; i--) {
          if (list.get(i).length() == tgt) {
               result += list.remove(i).length();      }
     }
     return result;
}
```

K. What is output by the following code? _____

Recall the `toString` of Maps:
`{key1=value1, key2=value2, ... keyN=valueN}`

```
TreeMap<Integer, String> mk = new TreeMap<>();
mk.put(4, "EE");
mk.put(0, "AI");
mk.put(4, "CS");
mk.put(mk.size(), mk.get(2));
System.out.print(mk);
```

L. What is the worst case order of the following method? _____

`LinkedList` is the Java `java.util.LinkedList` class. N = `list.size()`.

```
public static int methodL(LinkedList<Integer> list, int tgt) {
     int result = 0;
     Iterator<Integer> it = list.iterator();
     while (it.hasNext()) {
          int n = it.next();
          if (n < tgt)  {
               it.remove();
               result += n;
          }
     }
     return result;
}
```

M. A method implements the Radix sort algorithm _____
as shown in class and uses a base of 10. The method
takes 10 seconds to complete with an array of 250,000 elements with a max value of 1,000.
What is the expected running time for the method given an array of 1,000,000 elements and a max value of 10,000,000?

N. Given an array based list and a linked list with equal sizes, will binary search typically be faster with the array based list, the linked list, or will the time be roughly the same. Justify your answer with one sentence.

O. What is the order (Big O) of the following method? _____
N = `data.length`. `data` contains N distinct elements.
(no duplicates) None of the elements equal `null`. The length of each `String` is <= 10.
The mergesort method is not shown, but it uses the standard mergesort algorithm presented in class.

```
public static ArrayList<String> methodO(String[] data) {
     ArrayList<String> r = new ArrayList<>(data.length);
     for (int i = 0; i < data.length; i++) {
          r.add(data[i]);
          mergesort(r); // Assume standard mergesort.
     }
     return r;
}
```

P. What is output by the following code? _____

```
String sData = "COMPUTER";
Stack314<Character> st1 = new Stack314<>();
for (int i = 0; i < sData.length(); i++) {
     char ch = sData.charAt(i);
     if (ch > 'M')
          st1.push(ch);
}
while (!st1.isEmpty())
     System.out.print(st1.pop());
```

Q. A priority queue maintains elements _____
in order of the assigned priority. The highest priority items
are at the front of the queue and the lowest priority items are at the back. If a priority queue used a doubly linked list as its internal storage container, what is the expected order of the enqueue method?

R. The following values are inserted one at a time in the order shown (left to right) into an initially empty binary search tree using the simple algorithm demonstrated in class. Draw the resulting tree.

```
16, 4, 8, 0, 8, 12
```

S. What is the result of a post order traversal of the tree from part 1.R? _____

T. Consider the tree on page 12 of this exam. What is the height of the node that contains 5? _____

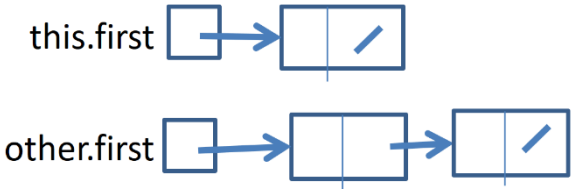
2. Linked Lists (22 points) - Complete the `listsConverge` instance method for the `LinkedList314` class. The method returns **`true`** if the calling object and a `LinkedList314` sent as a parameter ***converge***, **`false`** otherwise.

- **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 a reference to the first node in the chain of nodes. If the list is empty, first stores null.
- The last node in the chain of nodes has its `next` reference set to `null`.
- The list has a size instance variable that tracks the number of nodes (and elements) in the list.

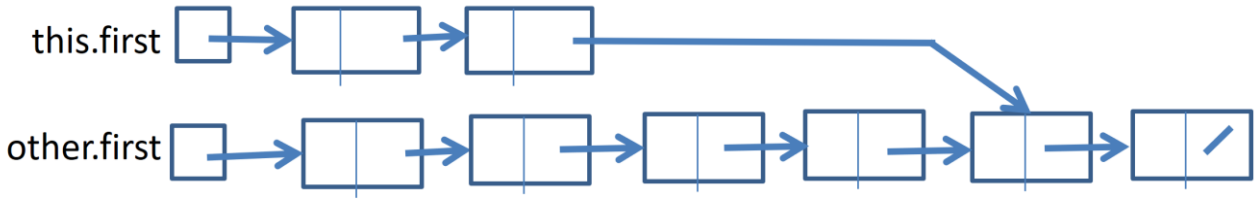
```
public class LinkedList314<E> {
      // Refers to first node in structure. Stores null iff size == 0
      private Node<E> first;
      private int size;

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

Two linked lists ***converge*** if at some point they refer to the same node and share the nodes after that point. In the examples the data stored by the lists is not shown. For example, the two lists to the right do not converge:



The two linked lists below converge. At some point they start sharing nodes.



Given the first example the method returns `false`. Given the second example the method returns `true`.

You may assume neither list has any cycles. A node's next reference never refers to a previous node in the structure.

You solution shall be O(1) space. Meaning no other data structures, no recursion.

Do no use any other Java classes or methods besides the nested Node class.

You must explain your approach and algorithm below. This is worth 3 points and will be graded based on the clarity and simplicity of your explanation and if it matches your actual code.

Explain your algorithm here: (3 points)

```
/* pre: other != null
   post: Per the problem description. this and other not altered.*/
public boolean listsConverge(LinkedList314<E> other) {
```

3. Maps (22 points) - Complete the `getTeams` method. The method accepts an array of `String`s. The length of the array shall be an even number. The `String`s are the names of NCAA women's volleyball teams and come in pairs. Each pair represents a game between the two teams. The team that won the game appears first in the pair and the team that lost the game appears second in the pair.

For example, given the array:

["UT", "Baylor", "UT", "A&M", "Baylor", "TCU"]

There are three games represented:

- UT defeated Baylor in the first game
- UT defeated A&M in the second game
- Baylor defeated TCU in the third game.

The method returns a `Map` whose keys are `String`s, the team names, and whose values are `Integer` objects. The `Integer` value is the difference between the number of games the team won and the number of games the team lost based on the data in the array of `String`s.

In the example UT has a difference of +2 (won 2, lost 0), Baylor had a difference of 0 (won 1, lost 1) and A&M and TCU both have a difference of -1 (won 0, lost 1).

The `getTeams` method also accepts an int parameter, `cutoff`.

Only teams with a win difference greater than or equal to the `cutoff` are included in the returned map.

So for example if the cutoff were 1 or 2, UT would be the only key in the map, if the cutoff were 0 then UT and Baylor would be keys in the returned map. If the cutoff were -1 or less than all four teams from the small example would be in the returned map.

The values in the returned map are sorted based on team names.

You may create `TreeMap`s and / or `HashMap`s.

You may use the following methods from the `Map` Interface: `V put(K, V), V get(Object), boolean containsKey(K), Set<K> keySet(), V remove(Object), V replace(K, V).`

You may create and use `Iterator` objects (implicit or explicit) for the set returned by the `keySet` method. You may create and use `Integer` objects.

Do not use any other Java classes or methods. Do not create any new native arrays.

Do not use recursion.

```
/* pre: games != null, games.length % 2 == 0.
   post: Per the problem description.
         games is not altered as a result of this method.*/
public static Map<String, Integer> getTeams(String[] games,
                                            int      cutoff) {
```

4. Binary Trees - (14 points.) Implement an instance method for a binary tree class that determines the number nodes in the tree whose stored values, when added together with the values stored in their children (if any), equal a target value.

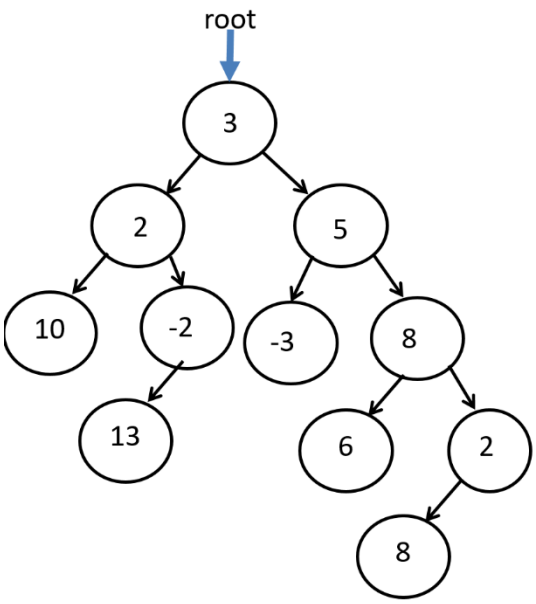
```
public class BinaryTree {

      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.



If the target value was 10 the method would return 5 based on the following nodes and their child nodes. The notation for the values is (node value, left child node value, right child node value). If a child does not exist it appears a hyphen.

`(3, 2, 5), (2, 10, -2), (10, -, -), (5, -3, 8), (2, 8, -)`

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.

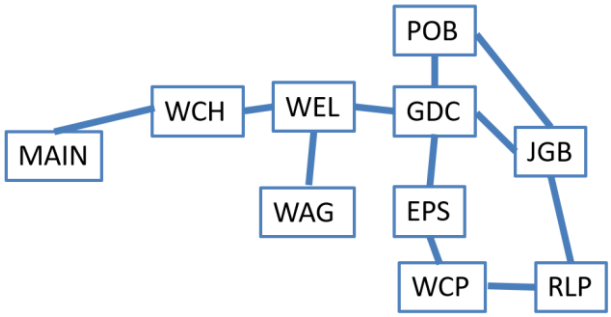
<u>You may add one helper method, but it cannot have more than 2 parameters.</u>

You may NOT add any class or instance variables to the BinaryTree or BNode classes.

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

5. Recursion (22 Points) Write a recursive backtracking method that determines if a path with a given number of links exists from one building to another.

In colder climates, buildings are often linked via tunnels or sky-bridges. Imagine if some of the buildings on campus were linked in such a way. Each rectangle represents a building. A direct link (tunnel or sky-bridge) exists between the buildings if a line connects the rectangles representing the buildings. If a link exists one may travel either way along the link.



Write a recursive backtracking method that returns `true` if a path exists from a given start building to a given end building with exactly a given number of links.
For this question the path may not visit the same building more than once. Starting in a building counts as visiting it.

Consider these examples where the start is **WAG** and the destination is **EPS**. -> represent one link in the path.

path length: 1 -> not possible
path length: 2 -> not possible
path length: 3 -> possible (WAG -> WEL -> GDC -> EPS)
path length: 4 -> not possible
path length: 5 -> not possible (Note, if we were allowed to visit buildings more than once we could go WAG -> WEL -> GDC -> EPS -> WCP -> EPS, but this is not allowed)
path length: 6 -> possible (WAG -> WEL -> GDC -> JGB -> RLP -> WCP -> EPS)
path length: 7 -> possible (WAG -> WEL -> GDC -> POB -> JGB -> RLP -> WCP -> EPS)

Note, if the start building and the end building are the same and the allowed path length is 0, then the method returns `true`. For example, if the start is GDC, the end is GDC, and the path length is 0, the method returns `true`.

The fourth parameter to the method is a `BuildingMap` object. This class handles many of the low level details of the buildings and their connections. It has the following public methods:

```
String[] connected(String building) return an array of Strings building is
connected to. The array will be length 0 if not connected to any buildings.

void setVisited(String building, boolean visited) set the visited property
for this building to the given value. Initially all buildings have their
visited property set to false.

boolean visitedStatus(String building) Return the visited status of building.
```

Complete the method on the next page. The `BuildingMap` must be restored to its original state (all buildings unvisited) by the time the method completes.

Your solution must use recursive back tracking.

Do not create any new data structures.

You may use the `BuildingMap` class, the arrays returned by the `BuildingMap` object, and `Strings` including the `String` `equals` method. Do not use any other Java classes or methods

```
/* pre: start != null, end != null, map != null, all buildings in map have
      their visited property set to false.
   post: per the problem description. map is unaltered by this method. */
public static boolean pathExists(String start, String end, int links,
                                 BuildingMap map) {
```