Points off	1	2	3	4	5	6	Total off	Net Score

CS 314 – Exam 2 – Fall 2018

Your Name: _____

Your UTEID: _____

Circle your TAs Name: *Amir* **Andrew** BEN Chris *ETHAN*
Hailey *JACOB* **LEVI** ***Lucas*** **Madhav**
Smruti Suhas

Instructions:

1. There are **6** 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 use a calculator or any other electronic devices 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 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, 15 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 returned by the method call `a(9)`? _____

```
public static int a(int n) {
     if (n <= 2)
          return n * 2;
     else if (n % 2 == 0)
          return 5 + a(n - 3);
     else
          return 2 + a(n - 1);
}
```

B. What is output by the following code? _____

```
System.out.print(b("A").length());

public static String b(String s) {
     if (s.length() <= 15) {
          System.out.print(s.length() + " ");
          String r = b(s + s + s + s.charAt(0));
          System.out.print(r.length() + " ");
          return r;
     }
     return s;
}
```

C. What is returned by the method call `c(6)`? _____

```
public static int c(int x) {
     if (x > 10)
          return 10;
     else
          return 5 + c(x + 3) + c(x + 1);
}
```

D. What is output by the following code? _____

```
System.out.print(d(5));

public static int d(int n) {
     if (n > 7)
          return 1;
     else
          return d(n - 1) + 1 + d(n + 1);
}
```

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

3 10 * 6 / 2 4 - * _____

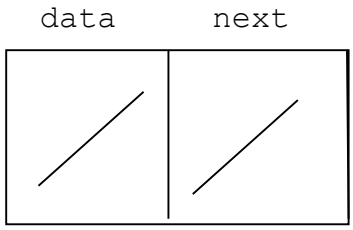
F. Given an array with 1,000,000 elements in random order, a method that uses the mergesort algorithm takes 100 seconds to complete. What is the expected time for the method to complete given an array with 2,000,000 elements in random order?

_____

G. In lecture we discussed five sorting algorithms: selection sort, insertion sort, radix sort, quicksort, and mergesort. Of these 5 sorts, which ones are stable?

_____

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`.




```
Node<Object> n1 = new Node<>("DOG", null); // params are (data, next)
Node<Object> n2 = new Node<>(n1, n1);
Node<Object> n3 = new Node<>(null, n2.next);
n3.data = n1.data;
```

Answer:

I. The following method takes 2 seconds to complete when `list.size()` = 1,000.
What is the expected time for the method to complete when `list.size()` = 3,000?

_____

```
public static int methodI(LinkedList<Integer> list) {
     int r = 0;
     for (int i = 0; i < list.size(); i++)
          for (int j = i + 1; j < list.size(); j++)
               if (list.get(i) == list.get(j))
                    r += list.get(i);
     return r;
}
```

J. What is output by the following code? The class is a traditional queue, not the Java Queue interface.

```
Queue314<Integer> q = new Queue314<>();
for (int i = 4; i <= 10; i++) {
     q.enqueue(i * 2);
     q.enqueue(i / 2);
}

for (int i = 0; i < 5; i++) {
     int x = q.front();
     q.dequeue();
     System.out.print(x + " ");
}
```

K. Is it more important for an array based list class or linked list class to implement the `Iterator` interface? Explain your answer:

L. What is the minimum number of nodes in a complete binary tree with a root that has a height of 4?

_____

M. What is the result of a pre order traversal of the tree to the right?

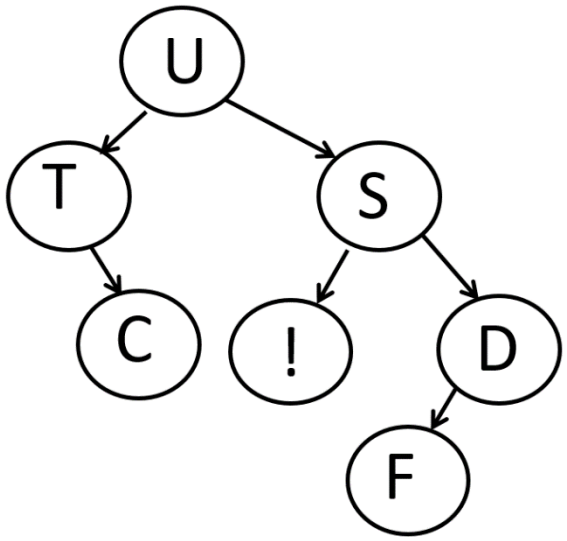
_____

N. What is the result of an in order traversal of the tree to the right?

_____

O. What is the result of a post order traversal of the tree to the right?

_____



2. Trees (9 points) Implement a helper method for a binary tree class that returns `true` if the tree is full, `false` otherwise. Recall, our definition for a full binary tree is one in which every node is a leaf (no child nodes) or has 2 child nodes. For this question an empty binary tree is full binary tree.

For this question, you may not add any helper methods.

```
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 store null if no child exists on that side
          private BNode left;
          private BNode right;
     }

     public boolean isComplete() {
          return helper(root);
     }

     private boolean helper(BNode n) {
          // complete this method below.
```

3. Maps (19 Points) Write a method that determines the *dot product* of two maps of word frequencies.

The keys of the maps are `Strings` and the associated value is an `Integer` that is the frequency of the `String` in a given text.

So given the phrase "to be or not to be" the map of word frequencies is :
`([to, 2], [be, 2], [or, 1], [not, 1])`
This is m1 in the example.
The actual order of the keys in the map could be different.

and given the phrase "doubt truth to be a liar" the map of word frequencies is:
`([doubt, 1], [truth, 1], [to, 1], [be, 1], [a, 1], [liar, 1])`
This is m2 in the example.
Again, the actual order of the keys in the map could be different.

$m1 \cdot m2$ is the *dot product* of two maps representing word frequencies.
This is defined as $\sum_{w_i} m1(w_i)m2\ (w_i)$ where

- w_i is the i^{th} word in the ***combined*** words of m1 and m2
- $m1(w_i)$ is the frequency of the i^{th} word in m1
- $m2(w_i)$ is the frequency of the i^{th} word in m2.

So in the example, the combined words of the two maps m1 and m2 are:
`[to, be, or, not, doubt, truth, a, liar]` and the dot product is:
`1*2 + 1*2 + 1*0 + 1*0 + 0*1 + 0*1 + 0*1 + 0*1 = 4`

Methods you may use from the `Map` interface:

```
V get(K key) return the value associated with the given key or null if
          given key is not present in this map

int size() the number of key value pairs in this Map

V put(K key, V val) associates the given val with the given key in
          this map. If key is already present replace and return the
          old value.

Set<K> keySet() returns a set of all keys in this Map.
```

You may also `for-each` loops and / or iterators.

Do not use any other Java classes or methods.

Your method shall be as efficient as possible given the restrictions of the question.

Complete the method on the next page.

```
/* pre: m1 != null, m2 != null
   post: returns the dot product of m1 and m2 as defined above.
     Neither m1 or m2 are altered by this method. */
public static int dotProduct(Map<String, Integer> m1,
                             Map<String, Integer> m2) {
```

4. **Linked Lists 1 (19 points)** - Complete the `removeFirstOccurrenceStartingAt` instance method for the `LinkedList314` class. The method removes the first occurrence of a given object that occurs at or after a given index. The method returns `true` if the list was altered as a result of this call, `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 list uses zero based indexing.
- The `LinkedList314` class uses singly linked nodes.
- **The list uses a header node. The `header` reference always refers to the header node.**
- **The header node does not store any data.**
- The list does not store `null` data values.
- 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 `Objects`.
- **You may not use any other Java classes or native arrays.**

```
public class LinkedList314<E> {

     private final Node<E> header = new Node<>();

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

Examples of calls to `removeFirstOccurrenceStartingAt(int start, E tgt)` on various lists. `removeFirstOccurrenceStartingAt` is abbreviated to `rFO` for these examples.

In these examples the lists contain `String` objects.
The header node is not shown in the abstract lists below.

```
[A, B, C, A, D].rFO(0, A) -> returns true, list becomes [B, C, A, D]

[A, B, C, A, D].rFO(1, A) -> returns true, list becomes [A, B, C, D]

[A, B, C, A, D].rFO(3, A) -> returns true, list becomes [A, B, C, D]

[A, B, C, A, D].rFO(4, A) -> returns false, list unchanged

[A, B, C, A, D].rFO(7, A) -> returns false, list unchanged

[A, B, C, A, D].rFO(0, X) -> returns false, list unchanged

[].rFO(5, X) -> returns false, list unchanged
```

```
/* pre: tgt != null, start >= 0
   post: Per the problem description. */
public boolean removeFirstOccurrenceStartingAt(int start, E tgt) {
```

5. **Linked Lists 2 (19 points)** - Complete the `containsAll` instance method for the `LinkedList314` class. The returns `true` if the calling list contains all of the elements in the list sent as a parameter. **Unlike the `isPermuation` method from the CodeCamp assignment, a single element in the calling list <u>can</u> match multiple elements in the other list.**

- **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.
- **When the list is empty, `first` stores `null`. (Different than previous question.)**
- The list does not store `null` data 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 `Objects`.
- **You may not use any other Java classes or native arrays.**

```
public class LinkedList314<E> {

     private Node<E> first;

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

Examples of calls to `containsAll(LinkedList314<E> other)` on various lists.

In these examples the lists contain `String` objects.

```
[].containsAll([A, B, C]) -> returns false

[].containsAll([]) -> returns true

[A, B, C].containsAll([]) -> returns true

[A, B, C].containsAll([A, A, C, B, A, B, C, A]) -> returns true

[A, A, C, B, A, B].containsAll([C, A, B]) -> returns true

[A, B, C].containsAll([X, A, C, B, A, B]) -> returns false

[A, B, C, D, E].containsAll([A, A, C, B, A, B, C, A]) -> returns true
```

```
/* pre: other != null, post: per the problem description.
     Neither list is altered by this method call. */
public boolean containsAll(LinkedList314<E> other) {
```

6. Recursion (19 Points) Write a recursive backtracking method that given an array of `Rectangle` objects, returns `true` if there is a subset of the `Rectangle`s that can fill another given rectangular region, false otherwise.

The small rectangles are not allowed to overlap in the larger rectangular area we are trying to fill.

A given rectangle can only be used once.

`Rectangle`s cannot be rotated. For example, if a given `Rectangle` has a width of 5 and height of 3, it must be placed so that it covers 3 rows and 5 columns. It cannot be rotated to cover 5 rows and 3 columns.

Consider the following example with a 7 by 3 region to fill and the following rectangles. Each rectangle has a given width and height. The width is the first value and the height is the second value.

```
               A       B       C       D       E       F       G
Rectangles: ([4, 1], [3, 3], [3, 2], [1, 2], [3, 4], [4, 5], [1, 1])
```

A	A	A	A	B	B	B
C	C	C	D	B	B	B
C	C	C	D	B	B	B

For the given example, there is a subset of rectangles that can fill the larger, 7 x 3 rectangular region.

Use the following `Rectangle` class:

```
public class Rectangle {
     public int width;
     public int height;}
```

You may use the following method to alter a 2d array of `boolean`s.

```
/* Set all elements in mat specified by the given Rectangle, placed such that
     r and c are the upper left corner of the Rectangle, to the given value.
     All cells specified must be inbounds. */
    private static void setVals(int r, int c, Rectangle rect,
            boolean val, boolean[][] mat) {
```

You may also use the `allMatch` method to check if all elements in a region of a 2d array equal a given value.

```
/* Return true if all elements in mat specified by the given Rectangle,
placed such that r and c are the upper left corner of the Recatngle, equal
the given value. All cells specified must be inbounds. */
private static boolean allMatch(int r, int c, Rectangle rect, boolean val,
                     boolean[][] mat) {
```

The helper method shall handle one Rectangle at a time.

Do not use any other Java classes or methods.

Do not add any other helper methods on this question.

```
/* rects != null, no elements of rects == null, goal != null
     All widths and heights of all Rectangles > 0.
     No Rectangles are altered by this method. */
public static boolean canFill(Rectangle[] rects, Rectangle goal) {
     boolean[][] mat = new boolean[goal.height][goal.width];
     return helper(0, rects, mat, goal);
}

/* i is the index of the only Rectangle to consider in this method.
   Returns true if we have found a subset of Rectangles from rects to
   cover all elements of mat without any overlap.  */
private static boolean helper(int i, Rectangle[] rects,
               boolean[][] mat, Rectangle goal) {
```