Points off	1	2	3	4	5		Total off

CS 314 – Exam 2 – Spring 2018

Your Name____________________________________

Your UTEID ___________________________________

Circle your TA's Name:	Aish	Anthony	Chris	Dayanny	Hailey
	Ivan	Jacob	Joseph	Lucas	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 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). Give the most restrictive, correct Big O function. (Closest without going under.)
   e. Assume `log₂(1,000) = 10` and `log₂(1,000,000) = 20.`

A. What is returned by the method call `a(8)`? _________________________

```
public static int a(int x) {
     if (x == 0)
          return 5;
     else
          return 3 + a(x - 3);
}
```

B. What is returned by the method call `b(4, 1)`? _______________________

```
public static int b(int x, int y) {
     if (x <= 0)
          return y;
     else
          return x % y + b(x - 1, y + 1);
}
```

C. What is returned by the method call `c("CS314")` ? _______________________

```
public static int c(String s) {
     if (s.length() == 0)
          return 314;
     else
          return s.length() + c(s.substring(1));
}
```

D. What is returned by the method call `d(7)`? _______________________

```
public static int d(int x) {
     if (x <= 1)
          return 3;
     else
          return x / 2 + d(x - 2) + d(x - 4);
}
```

E The following method takes 5 seconds to complete when N = 500,000. What is the expected time for the method to complete when N = 2,000,000?

____

```
public static double e(int N) {
     double r = 0.0;
     final int LIMIT = N / 2;
     for (int i = 0; i < LIMIT; i++)
          for (int j = Math.min(10_000, N); j >= 0; j--)
                r += (double) j / (i + 1);
     return r;
}
```

F. What is output by the following code? ____

```
HashMap<String, Integer> hm = new HashMap<>();
hm.put("UT", 12);
hm.put("WU", 17);
hm.put("SU", 15);
System.out.print(hm);
```

G. The following method takes 5 seconds to complete when `list.size() = 20,000`. What is the expected time for the method to complete when `list.size() = 40,000`?

____

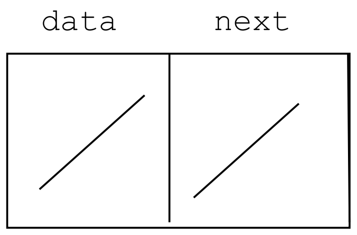
```
public static int g(LinkedList<Integer> list) {
     int t = 0;
     for (int i = 0; i < list.size(); i++) {
          int tgt = list.get(i);
          for (int j = i + 1; j < list.size(); j++) {
               if (list.get(j) == tgt)
                    t += list.get(j);
          }
     }
    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

H. The following method takes 20 seconds to complete when `list.size() = 1,000,000`. What is the expected time for the code to complete when `list.size() = 2,000,000`?

____

```
public static int h(ArrayList<Double> list) {
     int t = 0;
     for (int i = 0; i < list.size(); i++) {
          double a = list.get(i);
          for (int j = 0; j < list.size(); j++) {
               double b = list.get(j);
               for (int k = 1; k < list.size(); k *= 2)
                     if (a == list.get(k) && b == list.get(k))
                         t++;
          }
     }
     return t;
}
```

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

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

`50 5 10 - / 3 +` _________________

K. What is output by the following code? _____________________________________

```
Stack314<Integer> st = new Stack314<>();
for (int i = 1; i < 4; i++) {
     st.push(i / 2);
     st.push(i);
}
for (int i = 0; i < 6; i++) {
     st.pop();
}
while (!st.isEmpty()) {
     System.out.print(st.pop() + " ");
}
```

L. It takes a method using the selection sort algorithm 10 seconds to complete when sorting an array with 40,000 items in random order. What is the expected time for the method to complete when sorting an array with 120,000 items in random order?

____

M. A method uses the quicksort algorithm presented in class. It takes the method 100 seconds to complete in `case 1` below. What is the expected time for the method to complete in `case 2`?

____

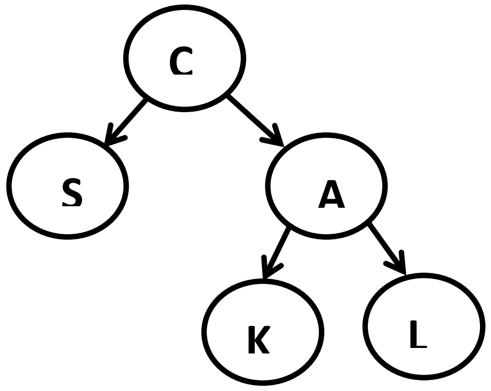
```
int[] m1 = new int[1_000_000];
quicksort(m1); // case 1
int[] m2 = new int[2_000_000];
quicksort(m2); // case 2
```

N. How many internal nodes are there in a complete binary tree with 9 nodes?

____

O. The following values are inserted 1 at a time into an initially empty binary search tree using the naive algorithm demonstrated in class. Draw the resulting tree.

```
5, 12, -6, 0, 12, 3, 5
```



P. What is the result of a preorder traversal of the above binary tree? ________________________

Q. What is the result of a postorder traversal of the above binary tree? ________________________

R. What is the order (Big O) of the following method? The `BST` class implements a binary search tree using the naive insertion algorithm demonstrated in class.

____

```
public static BST<Integer> q(int n) {
     BST<Integer> result = new BST<>();
     for (int i = 0; i < n; i++)
          result.add(i * 2);
     return result;
}
```

S. What is the order (Big O) of the following method? The `BST` class implements a binary search tree using the naive insertion algorithm demonstrated in class.

____

```
public static BST<Double> r(int n) {
     BST<Double> result = new BST<>();
     for (int i = 0; i < n; i++)
          result.add(Math.random());
     return result;
}
```

T. Why would an array based list class implement the `Iterable` interface when the method shown below is O(N) as written?

____

```
public static int t(ArrayList<Integer> list) {
     int r = 0;
     for (int i = 0; i < list.size(); i++) {
          r += list.get(i);
     }
     return r;
}
```

2. Linked Lists (20 points) - Complete the `getCopyWithoutTarget` instance method for the `LinkedList314` class. The method returns a copy of the calling object, except any values in the original list equal to a given target value are not included. The relative order of elements not equal to the target is the same between the result and the original 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`.
- 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, the `equals` method for `Object`s, and the default `LinkedList314` constructor.
- **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;

          public Node(E d) {  data = d; }
     }
}
```

Examples of calls to `getCopyWithoutTarget (E tgt)` on various lists. The initial list is on the left and the list returned by the method is on the right

In these examples the lists contain `Integer` objects.

```
[], tgt 5 -> returns []

[5], tgt 5 -> returns []

[5, 5, 1, 5, 5], tgt 5 -> returns [1]

[5, 5, 5, 5], tgt 5 -> returns []

[5, 7, 5, 3, 5, 3], tgt 5 -> returns [7, 3, 3]

[7, 7, 3, 12, 15, 0, -3], tgt 5 -> returns [7, 7, 3, 12, 15, 0, -3]
```

Complete the method on the next page.

```
/* pre: tgt != null
   post: Per the problem description. This list is not altered as a
     result of this method. */
public LinkedList314<E> getCopyWithoutTarget(E tgt) {
```

3. Queues (20 points) - Complete the `isStrictlyDecreasing(Queue314<Integer> q)` method. The method determines if the elements in a queue are strictly decreasing from front to back.

For elements to be strictly decreasing, each successive element must be less than the previous element.

All values in the initial queue are greater than 0.

Recall the queue methods:

```
public class Queue314 {

     public boolean isEmpty();

     // access front element, without removing, pre: !isEmpty()
     public E front();

     // remove front element, pre: !isEmpty()
     public E dequeue();

     // add val to the back of this queue
     public void enqueue(E val)
}
```

The method you are implementing is NOT part of the `Queue314` class.

Examples of queues and expected results if they are passed as an argument to `isStringlyDecreasing`.

```
front -> [] <- back, returns true

front -> [12] <- back, returns true

front -> [12, 10, 8] <- back, returns true

front -> [12, 10, 8, 8, 7] <- back, returns false

front -> [12, 20, 8] <- back, returns false
```

The queue is restored to its original state by the end of the method.

Do not use any other Java classes or methods except implicit use of the Integer class.

Do not create any other temporary data structures, not even another Queue. Do not use recursion.

Recall, all values in the initial queue are greater than 0.

Complete the method on the next page.

```
/* pre: q != null, all elements of q > 0
   post: return true if the elements in this queue are in strictly
     decreasing order, front to back, false otherwise.
     The queue is restored to its original state.       */
public static boolean isStrictlyDecreasing(Queue314<Integer> q) {
```

4. Maps (20 points) Write a method to determine the academic class with the highest average GPA. The method accepts one parameter a `Map<String, Map<String, Double>>` that represents students, the academic classes they have taken, and the grades earned in the classes. Consider the following example (quotes not shown on Strings).

```
Keys      Values (another map with Key-Value pairs
Mike      {CS312=2.67, CS314=2.33, REL372=2.0}
Lucas     {CS314=4.0, CS312=3.67, CS439=4.0, CS371m=4.0}
Chris     {CS331=3.67, CS312=4.0, CS439=3.33, CS314=3.0, M408D=3.0}
Shelby    {CHE302=3.3, CS312=4.0, CS331=3.0, CS439=3.0, CS371m=3.0}
```

In this example the method would return CS312 as the academic class with the highest average GPA, in this case 3.585. If there is a tie for highest average GPA you can return any of the academic classes that tie.

You may create and use another `Map` with `String`s as keys and `double[]` as values.

All of the values in the new, temporary map must be arrays of `double`s with lengths of 2.

Limit yourself to the following methods from the `Map` Interface:

V put(K, V)	adds a mapping from the given key to the given value if key already present, replaces old value with given value
V get(K)	returns the value mapped to the given key (null if none)
boolean containsKey(Object)	returns true if this key is present in the map, false otherwise
Set<K> keySet	returns a Set view of the keys in this Map

You may call the `iterator` method on the key set and use all of the methods from the `Iterator` interface.

Do not use any other Java classes or methods.

Complete the method on the next page.

```
/* pre: m != null, at least one entry with value with at least one class.
   post: Per the problem description. m is not altered as a result of this
     method. */
public static String highestGPA(Map<String, Map<String, Double>> m) {
```

5. Recursive Backtracking (20 points) Write a method that uses recursive backtracking to determine if it is possible to cover all discussion sections based on TA's and their available times. In other words, is there a schedule that covers all sections?

TA data is stored in a 2d array of `String` objects. The 2d array may be ragged, in other words each row is not guaranteed to have the same number of columns. The first element in each row is the TA's name. This followed by the times the TA is available. Times are expressed as `String`s such as "M1", meaning the TA is available Monday's 1 - 2 pm. Each TA will have at least one available time.

Here is an example of the TA data. All values shown are `String`s.

```
Dayanny, M10, T9
Aish, M1
Hailey, T9
Ivan, M1
```

If the sections to cover were M10, M1, M1, T9 there is a solution to this scheduling problem.
M10 -> Dayanny, M1 -> Aish, M1 -> Ivan, T9 -> Hailey.

If the sections to cover were M10, M1, T9, T9 there is no solution. Dayanny is the only TA for the Monday 10 section and Hailey is the only TA left for two Tuesday 9 sections. One TA cannot cover two sections, especially if they meet at the same time.

The sections times are also stored in a 2D array of `String`s. The number of rows is the same as the number of TAs. Each row has two columns. The first element is the section time. The second element is initially `null` indicating no one has been assigned to that section.

So for example:
```
"M1", null
"M10", null
"M1", null
"T9", null
```

Complete a recursive backtracking method that uses recursive backtracking to determine if it is possible to fill the schedule so that each section is covered by a TA and each TA is assigned to exactly one section.

Do no attempt to do any preprocessing or apply any heuristics in the original method.

**** Your helper method must handle exactly one TA at a time. ****

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

Do not create any new data structures.

Complete the method and its helper method on the next page.

```
/* pre: tas != null, no elements of tas == null,
     sections != null, sections.length == tas.length, all rows in
     sections have 2 columns, the second element in each row equals null.
   post: return true if there is a solution to this scheduling problem,
     false otherwise. tas is not altered as a result of this call.*/
public boolean canSchedule(String[][] tas, String[][] sections) {
```