General instructions for this assignment.

You will have up to **2 hours** to complete this timed assignment (a 15-minute buffer has been provided).

Please note: Timed exams will auto-submit when the assignment deadline is reached or the exam timer has expired. Here are some tips to help you manage your exam time:

- When you start your exam, make a note on your scratch paper of the time you started and roughly calculate the end time.
- Have a clock nearby so that you can easily check the time.
- At regular intervals, check the Coursera timer to see how much time is remaining.
- Keep in mind that, while ProctorU may try to assist you in determining the amount of time left, you must refer to the official Coursera timer.
- Once the exam time ends, Coursera will automatically submit your work. Anything you have entered into the exam will be submitted at that time.

You may use **as many pages** of scratch paper as you need to work out your solutions. You are allowed to access the scratch paper throughout the duration of the exam. The scratch paper must be blank at the time you begin your exam and destroyed at the end.

You will need to use **one of the following web browsers**: Chrome, Safari, Internet Explorer, Firefox

You **may** use the following resources:

- An online calculator or computer calculator.
- Website: https://docs.oracle.com/javase/8/docs/api/ (including anything linked from that site or having a URL that starts with that prefix)

Notes/Books: This is an open-note/book exam.

Question 1 (10 minutes)

For each of the scenarios below, select, from the given list, the data structure that best fits the description or best solves the data storage challenge. Note that not all of the given data structures will be used, and some may be used more than once.

Data structures:

A. List
B. Queue
C. Stack
D. Set
E. Map
F. Tree
G. Trie
H. Heap
I. Graph

Scenarios:

I. The data structure upon which apples grow: _____
II. Storing a reference table of birthdays of people in the class, to be accessed by name: _____
III. Collecting just the days of the year that have birthdays to celebrate: _____
IV. Encoding a street map for navigation (computing routes to give directions for how to get from one place to another): _____
V. Efficiently store the names of millions of people (one copy of each name if there are duplicates): _____
VI. Representation of the power grid of a country: _____

Question 2 (5 minutes)

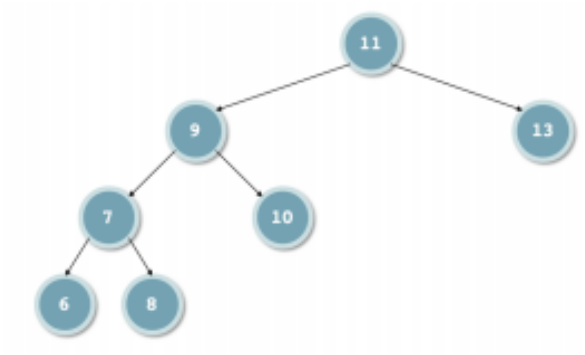
Among the following options, which best justifies the use of java.util.Vector instead of java.util.ArrayList?

A. It grows automatically as you insert elements.
B. Thread safety
C. Sortable
D. Cooler name
E. It doesn't matter, they are completely equivalent

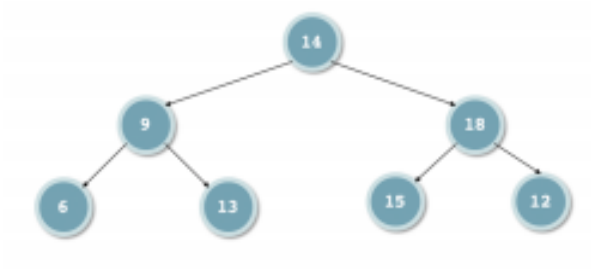
Question 3 (10 minutes)

Which of the following represents a valid AVL tree? Select all that apply.

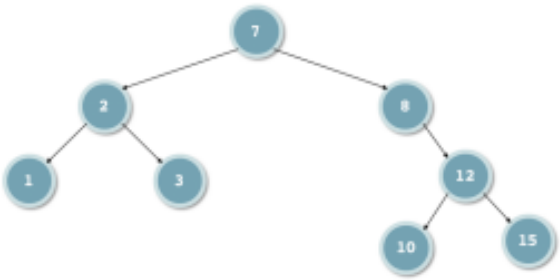
A.



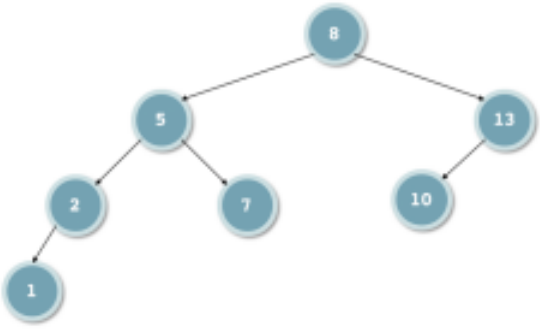
B.



C.



D.



Questi**on 4** (10 minutes)

Consider a HashSet using separate chaining with four buckets containing the following integer values:

- Bucket #0: 20 -> 4 -> null
- Bucket #1: null
- Bucket #2: 6 -> 16 -> null
- Bucket #3: 7 -> null

Given the values in the HashSet above, which of the following hash functions could be the one used in this implementation? Select the best answer.

A. hash(n) = n
B. hash(n) = n % 10
C. hash(n) = n * n
D. hash(n) = n * 10
E. None of the above

Question 5 (15 minutes)

```
1   public class MyClass {
2
3       private int g = 0;
4       private int k = 0;
5       private Object lock = new Object();
6
7       public void fun1() {
8           synchronized(lock) {
9               g += 3;
10          }
11          k++;
12      }
13
14      public void fun2(int a, int b) {
15          g += a;
16          a += b;
17          k = a;
18      }
19
20      public synchronized void fun3() {
21          g += k + 2;
22      }
23
24      public synchronized void fun4() {
25          k++;
26          g += 11;
27      }
28
29  }
```

For each of the 4 parts on the next page, assume that Threads T1 and T2 share the same MyClass object. Indicate whether each situation may or may not lead to a race condition, or whether it could never occur.

Question 5 (continued)

Part 1. T1 executes line 15 while T2 executes line 16.

A. This could occur, but would not lead to a race condition.
B. This could occur, and might lead to a race condition.
C. This could not occur.

Part 2. T1 executes line 26 while T2 executes line 21.

A. This could occur, but would not lead to a race condition.
B. This could occur, and might lead to a race condition.
C. This could not occur.

Part 3. T1 executes line 21 while T2 executes line 9.

A. This could occur, but would not lead to a race condition.
B. This could occur, and might lead to a race condition.
C. This could not occur.

Part 4. T1 executes line 11 while T2 executes line 25.

A. This could occur, but would not lead to a race condition.
B. This could occur, and might lead to a race condition.
C. This could not occur.

Question 6 (5 minutes)

There are bugs in this function. Fix them.

```
public static double mean(Collection<? extends Number> values) {
        double total = 0;

        if (values.isEmpty() || values == null)
                return Double.NaN;

        for (Number n : values) {
                total += n.doubleValue();
                if (n == null)
                        return Double.NaN;
        }

        return total / (double) values.size();
}
```

Enter the complete corrected function here:

```
public static double mean(Collection<? extends Number> values) {




}
```

Question 7 (25 minutes)

Consider the following function:

```
public static int count(int[] A, int[] C) {
        int M = A.length;
        int N = C.length;
        int count = 0;
        for (int i = 0; i < M; i++) {
                for (int j = 0; j < N; j++) {
                        if (A[i] > C[j])
                                count++;
                }
        }
        return count;
}
```

I. What is the asymptotic complexity as written? (4 minutes)

A. O(1)
B. O(N)
C. O(M+N)
D. O(M*N)
E. O(M^N)
F. May not terminate

II. Write a faster version (20 minutes):

III. What is the asymptotic complexity of an optimal implementation? (1 minute)

A. O(N)
B. O(M + N)
C. O(M * N)
D. O(M * log(M) + N * log(N))
E. O(M * log(M) * N * log(N))

Question 8 (20 minutes)

Write a function to reverse a linked list. The function should take a list node that is the head of the list and return a node that is the new head of the reversed list. This function should be destructive (it will necessarily alter the nodes in the list instead of copying the data stored therein). **You may not make any function or method calls at all from your function.** Anything slower than O(N) will be penalized.

```
public class LinkedList {
	public static class Node {
		String value;
		Node next = null;
	}

	public static Node reverse(Node head) {
		/* Insert code here */









	}
}
```

Question 9 (20 minutes)

In the programming assignments you were asked to implement a function to check if a graph walk was a Hamiltonian Cycle for a given graph:

isHamiltonianCycle: Given a Graph and a List<String> of node values, this method indicates whether the List represents a Hamiltonian Cycle through the Graph.

A Hamiltonian Cycle is a valid path through a graph in which every node in the graph is visited exactly once, except for the start node, which must be visited twice and must be identical to the end node, so that the path forms a complete cycle.

If the values in the input List represent a Hamiltonian Cycle given the order in which they appear in the List, the method should return true, but the method should return false otherwise, e.g., if the path is not a cycle, if some nodes are not visited, if some nodes are visited more than once, if some values do not have corresponding nodes in the graph, if the input is not a valid path (i.e., there is a sequence of nodes in the List that are not connected by an edge), etc.

The method should also return false if the input Graph or List is null.

A. Do each of the following tests give you useful information (even if it's only slightly useful or might not be a test you would actually choose to use) towards determining whether the walk is a Hamiltonian cycle or not? Answer true or false for each test. Consider each test independently from the others.

    a. Are there any duplicates in the list other than the first and last nodes? (T / F)
    b. Is there at least one node in the graph with at least two incoming edges? (T / F)
    c. Are the first and last nodes in the list the same? (T / F)
    d. Is the graph a directed graph? (T / F)
    e. Is the input list a null pointer? (T / F)
    f. Are all of the nodes in the graph in the list? (T / F)
    g. Is the graph a tree? (T / F)
    h. Is the input graph actually a null pointer? (T / F)
    i. Is there a repeated edge in the list? (T / F)
    j. Is the list the shortest walk (with the same start and end node) that traverses all nodes in the list? (T / F)
    k. Does the list contain null values? (T / F)
    l. Is the length of the list the same as the number of nodes in the graph? (T / F)
    m. Are all edges in the list valid edges in the graph? (T / F)
    n. Are all edges in the graph in the list? (T / F)
    o. Is the graph fully connected? (T / F)
    p. Are all items in the list nodes in the graph? (T / F)

B. What is the optimal runtime for this method (V = the number of nodes/vertices, E = the number of edges)? You may assume the graph uses HashMap<String, ? extends Collection<Edge>> to store nodes and edges, and that all HashMap operations are constant time.
    a. O(-V)
    b. O(1)
    c. O(V)
    d. O(V + E)
    e. O(V * E)
    f. Exponential
    g. Not guaranteed to terminate