This is the final examination for
CSC212: Data Structures
as taught by R. Jordan Crouser and Nicholas R. Howe in Spring 2022.

The following materials are **permitted** while taking this examination:

- a single 8.5x11 sheet of paper (double-sided) containing your own handwritten or typed notes
- blank scratch paper (provided at the end of this packet)

Honor code: no other resources are permitted during this exam.
This includes (but is not limited to): textbooks, online materials, tutors, teaching assistants, and other students.

If you encounter any issues while taking this exam,
the instructors can be reached on Slack:
`@Jordan Crouser` and `@Nicholas Howe`

NAME: ____

SCORE: _________ **out of 58**

Question 0. Getting in the Groove (0 points)

Note: This question is optional, but strongly recommended.

Educational research studies[1,2] have suggested that people perform better on tests when they spend a few minutes thinking about things they're good at before they begin.

In the space below, briefly tell us about a time when you were **really successful** at doing something challenging (it doesn't have to be related to this course). If you prefer, you can draw a picture instead of writing.

[1] Lang, Jonas WB, and Jessica Lang. "Priming competence diminishes the link between cognitive test anxiety and test performance: Implications for the interpretation of test scores." *Psychological Science* 21.6 (2010): 811-819.

[2] Barrows, Jennifer, Samantha Dunn, and Carrie A. Lloyd. "Anxiety, self-efficacy, and college exam grades." *Universal Journal of Educational Research* 1.3 (2013): 204-208.

Question 1. Vocabulary (6 points)

Word Bank:

abstract call signature declare exception generic inheritance
initialize instance interface iterator method overload
override public recursion static type void

Fill in the blank with the term or concept that matches each of the definitions below:

(a) A class, interface, or method that operates on a parameterized type (e.g. `ArrayList<T>`) is called ______________________________.

(b) A(n) ______________________________ is a specialized object used to traverse or retrieve a Collection or Stream object's elements one by one.

(c) Java allows us to ______________________________ methods defined within a class, which means we can define multiple methods with the same name (as long as they have different argument lists).

(d) When a method in a subclass has the same name, same parameters or signature, and same return type (or sub-type) as a method in its superclass, then the method in the subclass is said to ______________________________ the method in the superclass.

(e) In Java, space on the heap is allocated when we ______________________________ an object.

(f) In an architecture diagram, we list the ______________________________ attributes and methods for each class, as well as indicate the relationships between the classes.

Question 2. Tracing Java Programs (6 points)

Consider the following program, which is made up of 4 files:

`MyInterface.java`:

```
1  interface MyInterface {
2    public String modify(String s);
3  }
```

`Main.java`:

```
1  class Main {
2    public static void main(String[] args) {
3      MyClass m = new MyOtherClass();
4      System.out.println(m.modify(args[0]));
5    }
6  }
```

`MyClass.java`:

```
1  class MyClass implements MyInterface {
2    public String modify(String s) {
3      return s.toUpperCase() + "!!!";
4    }
5  }
```

`MyOtherClass.java`:

```
1  class MyOtherClass extends MyClass {
2    public String modify(String s) {
3      s = super.modify(s);
4      return "***" + s + "***";
5    }
6  }
```

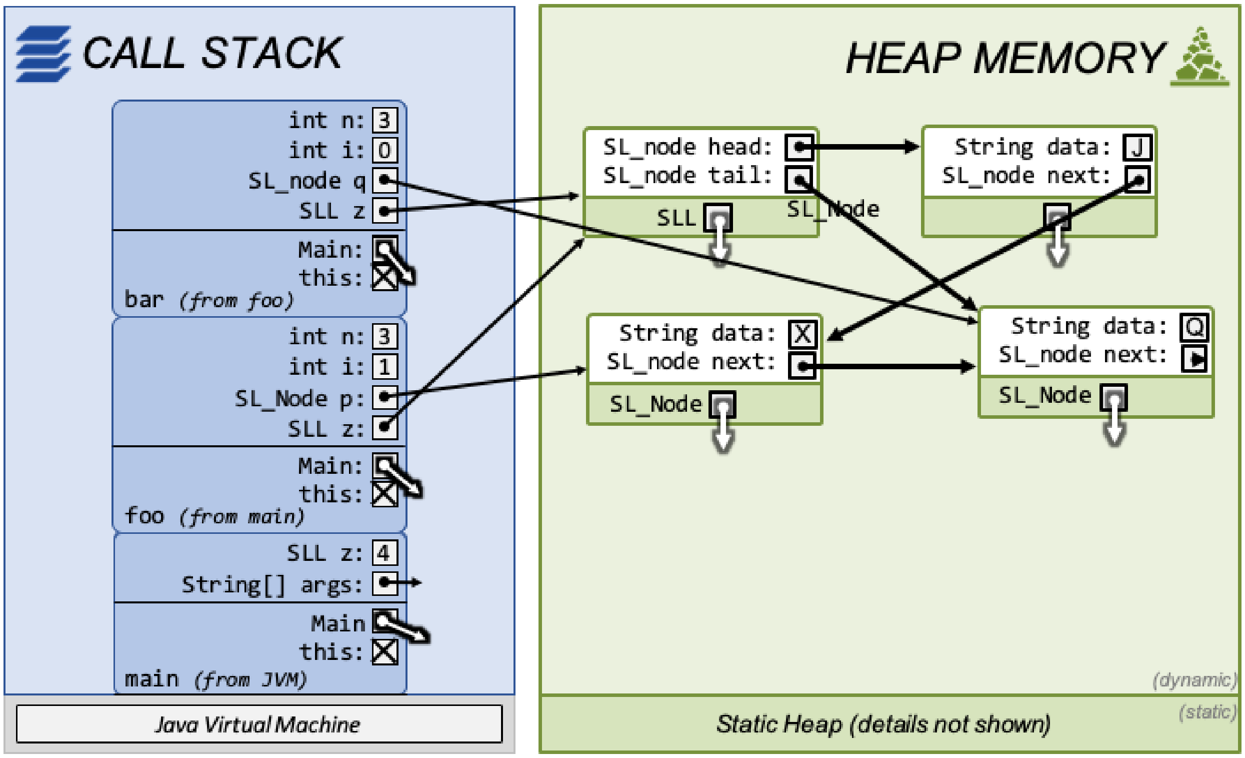
(a) What output is printed to the console if we run `java Main "almost done"`?

(b) What happens if we remove `implements MyInterface` on line 1 of `MyClass.java`?

(c) What happens if we remove `extends MyClass` on line 1 of `MyOtherClass.java`?

Question 3. Reading Memory Diagrams (8 points)

Consider the following diagram, depicting the state of the computer's memory during the execution of a program:



(a) What is the value of `i` in the currently executing method?

(b) What is the sequence of function calls that led to the situation shown?

...this question continued on next page.

(c) Suppose that the current function executes `q.setData(“R”)` and then returns. Will this cause any changes visible to the calling function?

(d) What value would this expression evaluate to in the current context?
`z.getHead().getNext().getData()`

Question 4. (Flawed) Operations on Lists (8 points)

Consider the list method implementations shown below. Each one has at least one bug, in the form of a special case that is not handled properly.

Example:

```
/** Inserts the given item at the head of the SLL */
 public void addFirst(T item) {
   head = new NodeSL<T>(item,head);
 }
```

What's wrong? *This method doesn't update the tail when adding to an empty list.*

For each of the following methods, describe the situation that is not properly handled.

(a)

```
/** Inserts the given item at the tail of the SLL */
 public void addLast(T item) {
     tail.setNext(new NodeSL<T>(item,null));
     tail = tail.getNext();
 }
```

What's wrong? ____

(b)

```
/** Removes the given item from the head of the list
 * @return v item removed */
 public T removeFirst() {
   T result = null;
   if (head == null) {
     throw new MissingElementException();
   } else {
     result = head.getData();
     head = head.getNext();
   }
   return result;
 }
```

What's wrong? ____

...this question continued on next page.

(c)

```
/** Inserts the given item in the SLL after the here node */
  public void addAfter(NodeSL<T> here, T item) {
    if (here == null) {
      // null means put at the head
      addFirst(v);
    } else {
      here.setNext(new NodeSL<T>(v,here.getNext()));
    }
  }
```

What's wrong? ____

(d)

```
/** Appends a list by transferring elements */
  public void appendTransfer(SLL<T> suffix) {
    tail.setNext(suffix.head);
    tail = suffix.tail;
    suffix.head = suffix.tail = null; // transfer
  }
```

What's wrong? ____

Question 5. Sorting (12 points)

The items below each show some of the steps of a particular sorting algorithm working on an array. The first line shows the state of the array at some point in the middle of the sort. The following lines show how that state changes as the algorithm works. Unless otherwise specificed, each row shows the next consecutive state after a swap operation.

Fill in **the next three rows** for each algorithm.

(a) SelectionSort

1	2	3	5	8	4	6	7
1	2	3	4	8	5	6	7

(b) InsertionSort

2	5	4	8	3	1	7	6
2	4	5	8	3	1	7	6
2	4	5	3	8	1	7	6

(c) MergeSort*

5	2	8	4	3	1	7	6
2	5	8	4	3	1	7	6
2	5	4	8	3	1	7	6
2	4	5	8	3	1	7	6
2	4	5	8	1	3	7	6

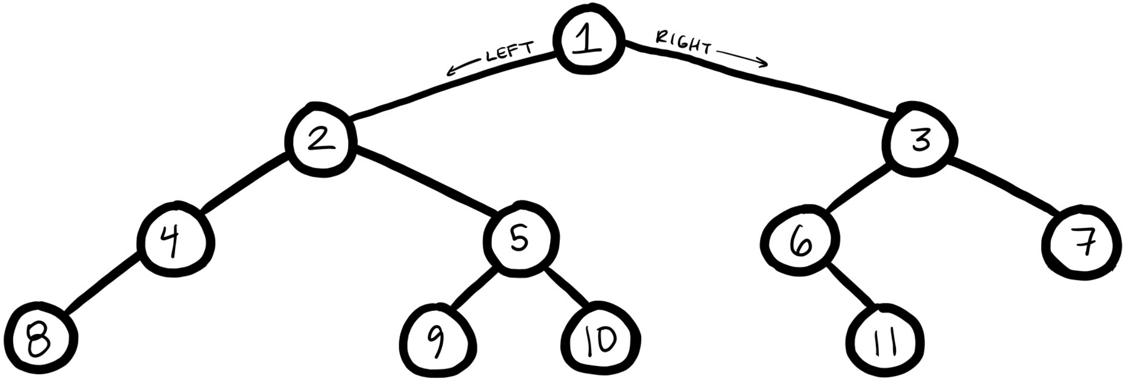
*here we show the result of each sorted merge

(d) HeapSort

5	2	4	6	3	1	7	8
5	6	4	2	3	1	7	8
6	5	4	2	3	1	7	8
6	5	7	2	3	1	4	8

Question 6. Tree Traversal (6 points)

Consider the following binary tree:



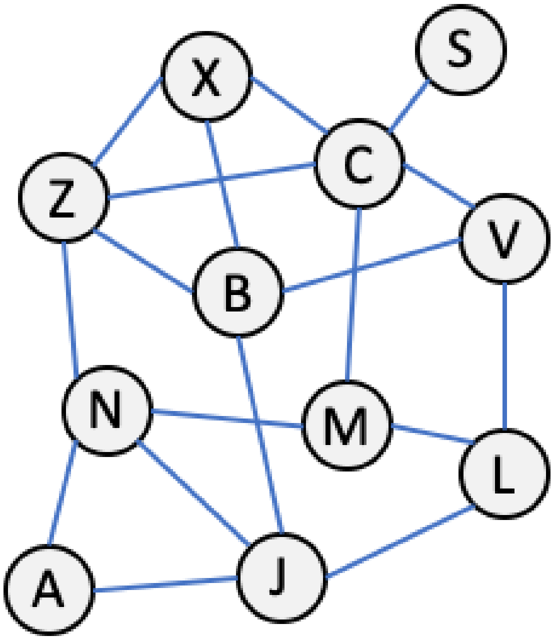
(a) In what order will the nodes be visited if we perform an **preorder** traversal?

(b) In what order will the nodes be visited if we perform an **inorder** traversal?

(c) In what order will the nodes be visited if we perform an **postorder** traversal?

Question 7. Graphs (12 points)

Consider the undirected graph shown to the right:



(a) Which node(s) could be the second to be visited (after L itself) when performing a breadth-first traversal beginning at node L?

(b) Which node(s) could be the second to be visited (after L itself) when performing a depth-first traversal beginning at node L?

...this question continued on next page.

(c) Which node(s) could be the third to be visited when performing a breadth-first traversal beginning at node L?

(d) Which node(s) could be the third to be visited (after L itself) when performing a depth-first traversal beginning at node L?

(e) When performing a breadth-first traversal beginning at node A, which node(s) could be the last visited?

(f) Assuming that all edges cost the same amount to traverse, what nodes lie on the least expensive path from X to M?

This page intentionally left blank as scratch paper.

This page intentionally left blank as scratch paper.

This page intentionally left blank as scratch paper.