CSC 210: Data Structures

Fall 2023 Midterm Exam

Due: Wednesday, October 10 at 11:59 pm (on Moodle)

- This is a take-home exam with unlimited time from when it is out to when it is due. We estimate that you should spend no more than two hours completing the exam.
- You will turn in your exam by scanning or photographing it and submitting on Moodle.
- The exam is open book, so you may use all notes and course materials. If you use any online resources besides the course web page, Moodle, the textbook, and the standard online Javadoc pages, please cite them explicitly below.
- You may not communicate or consult about the exam with anyone other than the professors.
- In the event that part of a question needs clarification or modification, a message will be posted on the general channel of the course slack workspace.
- If you are unable to make progress on any part of the exam, tell me what you know and what you tried: describe your thought process. Partial credit may be possible.

Please copy, sign, and date the statement below: “I have read the instructions above and certify that my work on this exam adheres to the Smith Honor Code. I have explicitly cited any resources used beyond those listed above.”

_____
_____
_____
_____
_____

Signature: _____

Citations. List any online citations below (page title & URL), or write “None used.”

Arrays (12 Points)

For the questions below, consider the following table of data sizes:

Size	Type
1 byte	**`boolean`**, **`byte`**
2 bytes	**`short`**, **`char`**
4 bytes	**`int`**, **`float`**
8 bytes	**`long`**, **`double`**, object reference

a. How many bytes of memory will be allocated by this line: **`int[] a = new int[20];`**

b. How many bytes of memory will be allocated by this line: **`String[] b = new String[7];`**

c. How many bytes of memory will be allocated by this line: **`Integer[] c = new Integer[12];`**

d. If an array **`d`** of float values is stored in the memory between address 6000 and 6200, how many values can the array store?

e. If an array **`e`** of double values is stored in the memory starting at address 8000, what would be the address where **`e[10]`** would be stored?

f. If an array **`f`** of boolean values is stored in the memory starting at address 8000, what would be the address where **`f[10]`** would be stored?

Vocabulary (16 points)

Please write 2-3 sentences that concisely and accurately identify (i) what the two terms below have in common, and (ii) what distinguishes them from each other.

a. Abstract data type (ADT) vs. a class implementing a data structure in Java

b. Singly linked list vs. doubly linked list

c. Iterator vs. index

d. Java built-in array vs ArrayList

Data Structure Selection (8 points)

We have talked on several occasions about the relative speed of different operations using different data structures. Thinking about the following in terms of time complexity, for each task, circle which data structure you would choose, and provide a justification for why. If the two are equally good, you can circle “either”.

a. Finding your name on a class roster where names are not ordered
   It would be faster if the roster were a(n) (circle one):

   Array Linked List Either

   Why?

b. Realizing you forgot the number “0” when trying to write down the numbers from 0 to 100
   It would be easier to fix if the number were in a(n) (circle one):

   Array Linked List Either

   Why?

c. Change the value at the current position while iterating.
   It would be faster when iterating on a(n):

   Array Linked List Either

   Why?

d. Store a playlist where you are trying to make sure there are smooth transitions between songs. If you change one song in the list, it might require changes to the songs before and after it.

   Singly Linked List Doubly Linked List Either

   Why?

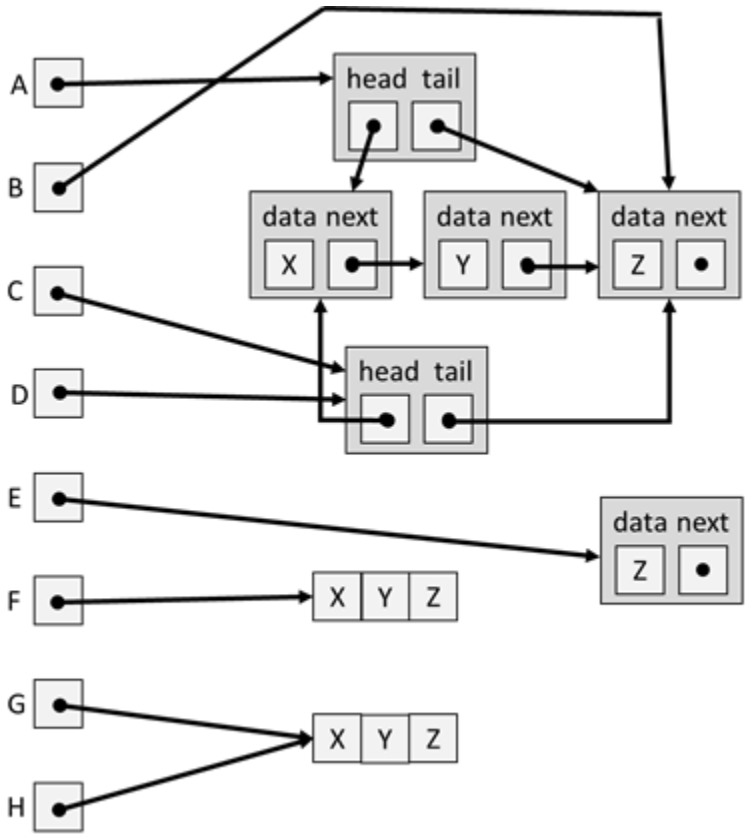
Copy Types (16 Points)

Consider the diagram of structures in memory below. For the eight variables A-H on the left-hand side, please identify the following groups. (Note that variables of several different types are mixed together in this problem, and by definition only variables of the same type can be copies of each other.)

a. Which variables are reference copies of each other?

b. Which variables are deep copies of each other?

c. Which variables are shallow copies of each other?



Iterators (10 Points)

Assume you have a singly-linked list *list,* as indicated here:

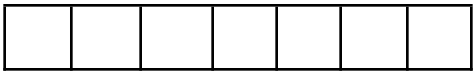
Assume this SLL class has the same methods you developed in Assignment 3.

a. Below, indicate each position that a list iterator can occupy as it moves through the list.

b. Assume you initialize a list iterator, named *pos*, and want to use it to traverse the list. Where indicated below, fill in the (1) values of each item in the list, and (2) indicate the position of *pos*:

i. `ListIterator<String> pos = new list.listIterator();`

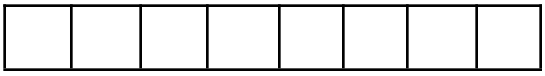
ii. `pos.add("S");`



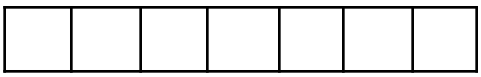
`pos.next();`

`pos.next();`

iii. `pos.add("T");`



iv. `pos.remove();`



Linked Lists (20 Points)

Below you will see some code that contains methods a through i, and some pictures labeled P through Z. Your job is to match each method with a picture that completely and accurately represents the structure(s) it sets up in memory. (Objects that will be garbage collected are not shown.) Not all code will necessarily have a picture, and not all pictures will necessarily correspond to a piece of code. Some pictures may correspond to more than one piece of code. For each, write the letter or letters you think are matches, or else write "None" if you think a particular item doesn't match anything on the other side.
Note that the code does compile and run without error (the main method is not shown).

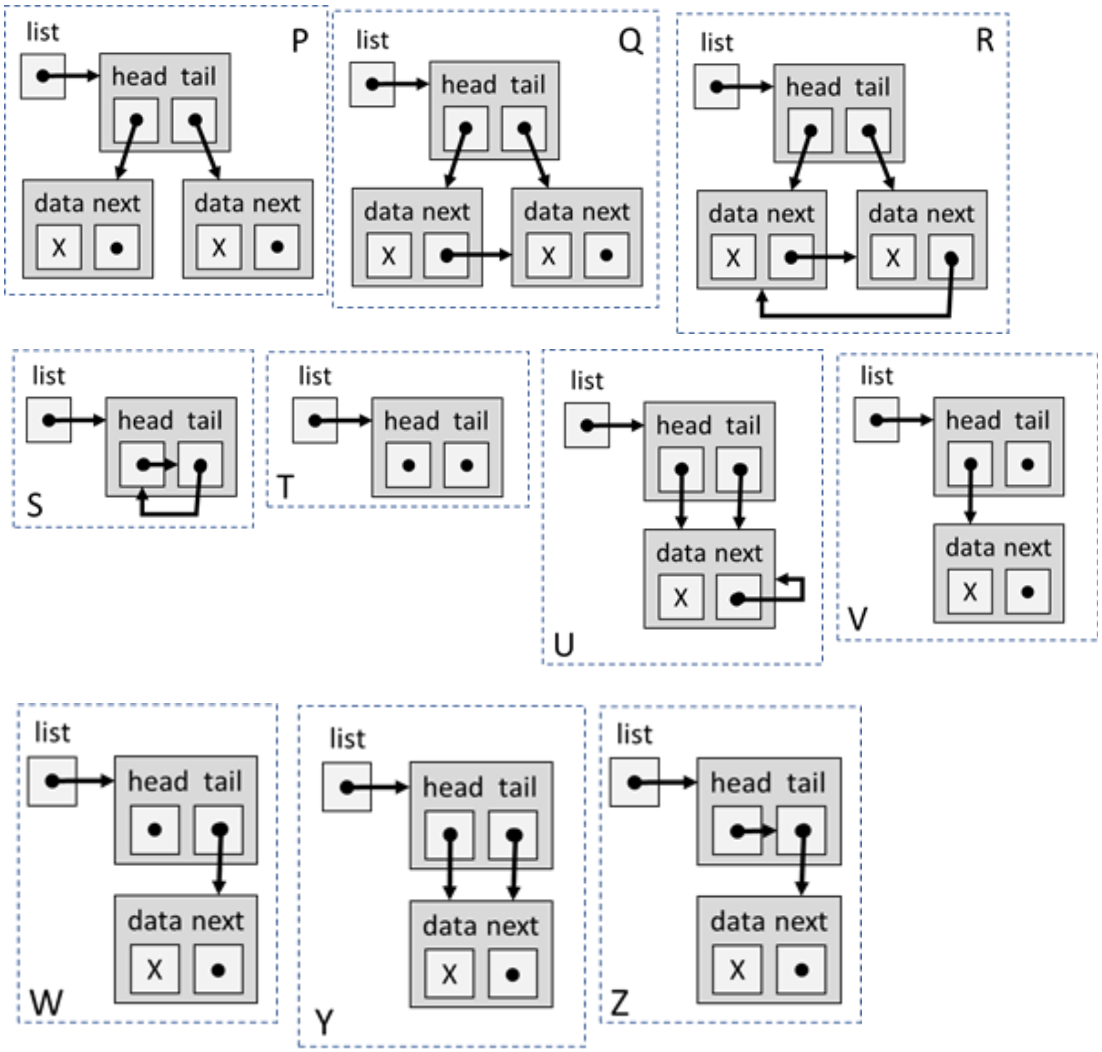
Method A: _____
Method B: _____
Method C: _____
Method D: _____
Method E: _____
Method F: _____
Method G: _____
Method H: _____
Method I: _____
Method J: _____

Picture P: _____
Picture Q: _____
Picture R: _____
Picture S: _____
Picture T: _____
Picture U: _____
Picture V: _____
Picture W: _____
Picture Y: _____
Picture Z: _____

```
class List {
    Node head;
    Node tail;

    List(Node h, Node t) {
        head = h;
        tail = t;
    }
}

class Node {
    char data;
    Node next;

    Node(char c, Node n) {
        data = c;
        next = n;
    }
}
```

```
public class Question{
    public void a() {
        List list = new List(null, null);
    }

    public void b() {
        Node n = new Node('X', null);
        List list = new List(n, n);
    }

    public void c() {
        List list = new List(null, null);
        list.head = list.tail = new Node('X', null);
    }

    public void d() {
        List list = new List(new Node('X', null), null);
        list.tail = list.head;
    }

    public void e() {
        List list = new List(new Node('X', null), new Node('X', null));
    }

    public void f() {
        List list = new List(null, null);
        list.head = new Node('X', list.tail);
        list.tail = new Node('X', list.head);
    }

    public void g() {
        List list = new List(null, null);
        list.head = new Node('X', list.head);
        list.tail = new Node('X', list.tail);
    }

    public void h() {
        List list = new List(new Node('X', null), new Node('X', null));
        list.head.next = list.tail;
```

```
    }

    public void i() {
        List list = new List(new Node('X', null), new Node('X', null));
        list.head.next = list.tail;
        list.tail.next = list.head;
    }

    public void j() {
        List list = new List(new Node('X', null), new Node('X', null));
        list.head = list.tail;
        list.tail = list.head;
    }
}
```



Loop Invariants (6 Points)

Consider the method definition below, and the statements that follow. Match each statement to its role in the loops. Each role should be matched to a statement, and each statement will be used exactly once.

```
/** Returns the k maximal elements in the array */
public static float[] largest(float[] arr, int k) {
    float[] largest = new float[k];
    for (int h = 0; h < k; h++) {
        largest[h] = Float.NEGATIVE_INFINITY;
    }
    for (int i = 0; i < arr.length; i++) {
        float item = arr[i];
        for (int j = 0; j < k; j++) {
            if ((item > largest[j])) {
                float temp = largest[j];
                largest[j] = item;
                item = temp;
            }
        }
    }
    return largest;
}
```

Statements:

I. The values in **largest[0]** through **largest[j-1]** are bigger than the value in **item**
II. The array largest contains the **k** biggest values anywhere in array **arr**
III. The array largest contains the **k** biggest values found in **arr[0]** through **arr[i-1]**
IV. None of the values from array **arr** are in the array **largest**
V. It is unknown whether the value of **item** is greater or smaller than any elements of **largest**
VI. All the values in **largest** are greater than or equal to the value in **item**

Match the statements above to the following role:

a. Initial conditions for the i loop _____
b. Initial conditions for the j loop _____
c. Desired outcome of i loop _____
d. Desired outcome of j loop _____
e. Loop invariant for the i loop _____
f. Loop invariant for the j loop _____

Blank page for showing work