Name: _____

CSC 210: Data Structures

Spring 2026 Midterm Exam

Section 01 (Prof. Rando)

Self-scheduled exam, Feb. 20 – 22, 2026

- You have 90 minutes to complete this exam, unless you have a documented accommodation for additional time. We anticipate that it will take less than one hour.
- The exam is closed book. You are allowed one 8.5”x11” page of notes (double-sided is ok).
- If you think there is a problem with the exam, do your best to interpret what is requested and document your choice with a note. Showing your work helps us give partial credit.
- You may not communicate or consult about the exam with anyone other than the professor until after the weekend is over.
- If you are unable to make progress on a part of the exam, describe your thought process: write down what you know and what you tried. Partial credit may be possible.
- All questions are weighted equally towards your final exam grade for the semester.

Please copy, sign, and date the statement below: “I have read the instructions above and certify that I have adhered to the Smith Honor Code while completing this exam.”

_____

_____

_____

_____

_____

Signature: _____

Abstract Datatypes

For each of the following operations, indicate whether the operation is part of the abstract datatype (ADT) for a list, or characteristic of a specific implementation of a list. Pick a checkbox and then provide a short explanation.

1. To find the last element in the list, you must visit every other element first.

   ☐ ADT ☐ Implementation

   Why?

2. It is very fast -- i.e., constant time / O(1) -- to get an element from a list.

   ☐ ADT ☐ Implementation

   Why?

3. Objects should stay in the list unless they are explicitly removed.

   ☐ ADT ☐ Implementation

   Why?

4. Adding an element at the beginning of the list is faster than adding at the end.

   ☐ ADT ☐ Implementation

   Why?

5. Elements have an order and will stay in that order.

   ☐ ADT ☐ Implementation

   Why?

6. An iterator has an operation that checks if there is another item remaining in a collection of elements.

   ☐ ADT ☐ Implementation

   Why?

Arrays

Suppose we have defined a DynamicArray class (like in assignment A1) that implements the interface below:

```
public interface DynamicArrayADT<T> {
    public T set(int index, T value);
    public T get(int index);
    public int size();
    public void add(int index, T value);
    public void add(T value);
    public T remove(int index);
}
```

For each question, you should assume that the class is backed by a standard Java array, and we have an instance **darr** whose starting configuration looks like this:

C	S	C	2	1	0

What will be the result of the following operations? Draw what the DynamicArray will look like and/or any exception that will be thrown.

// Part 1

```
darr.set(3, 'X');
```

// Part 2

```
darr.set(6, 'Q');
```

```
// Part 3

        darr.remove(1);



// Part 4

        darr.add(1, 'B');



// Part 5

        darr.add(1, 'B');

        darr.set(6, 'Q');



// Part 6
```

Based on your experience with DynamicArray (Assignment 1), do you have a question about the implementation that might affect how you draw the arrays above?

Linked Lists

In which of the following scenarios would a linked list offer a clear advantage over a base-type array (T[ ])?

If you have enough information to decide, check "**Yes**" (LL has an advantage) or "**No**" (No advantage to LL). Otherwise, check "**Unclear**" and write the question you would ask to help you decide.

1. You frequently insert new elements at the beginning of the data structure.

   ☐ Yes (Advantage) ☐ No (No Advantage)

   ☐ Unclear_____?

2. You frequently insert or delete elements from the middle of the data structure.

   ☐ Yes (Advantage) ☐ No (No Advantage)

   ☐ Unclear_____?

3. You frequently access items by index (e.g., "get the i^{th} item").

   ☐ Yes (Advantage) ☐ No (No Advantage)

   ☐ Unclear_____?

4. The total number of elements changes often and is difficult to predict.

   ☐ Yes (Advantage) ☐ No (No Advantage)

   ☐ Unclear_____?

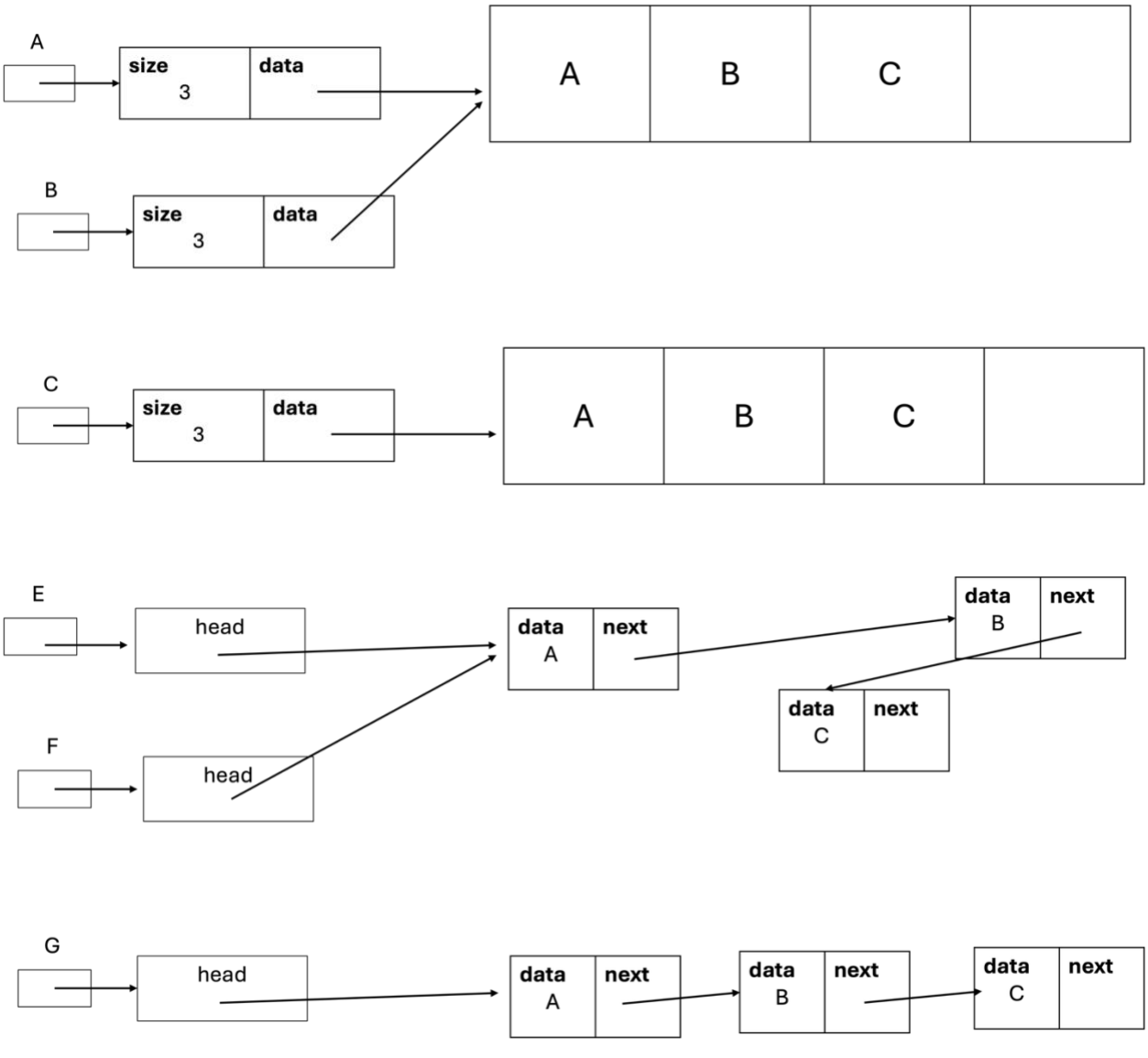
5. You need to traverse the structure sequentially, one node at a time.

   ☐ Yes (Advantage) ☐ No (No Advantage)

   ☐ Unclear_____?

Copy Depth

Imagine you have the following collection of objects:



1. Does A have any deep copies? List them.

2. Does G have any deep copies? List them.

3. Does F have any shallow copies? List them.

4. Does B have any shallow copies? List them.

5. Is C a deep copy of G? Why or why not?

Iterators

1. Imagine you are writing a loop to iterate over an array `T[ ]` called *myArray.* You might format this as:

```
i=0;
while (i < myArray.length){
  System.out.println(myArray[i]);
  i++;
}
```

   Here, when we reach `i++`, what kind of step is this?

   A. following a stored reference

   B. computing the next position by arithmetic

   C. searching from the beginning

   D. allocating a new element

Briefly explain your choice: _____

_____

_____

2. Similarly, now imagine you are looping over a Linked List (SLL) *myList*:

```
NodeSL cur = myList.head;
while (cur != null){
  System.out.println(cur.data);
  cur = cur.next;
}
```

   Here, when we reach `cur = cur.next`, what kindn of step is this?

   A. following a stored reference

   B. computing the next position by arithmetic

   C. searching from the beginning

   D. allocating a new element

Briefly explain your choice: _____

_____

3. Assume you have a list of elements called *myList*, as shown here. You initialize an iterator on *myList* and call it *pos.*

    a. What position will *pos* occupy right after initialization? Please draw it below using a ^ like we did in class.

    b. Now you call:

```
pos.next(); pos.next(); pos.next();
```

    Where is *pos* located now? Please draw it below using a ^.

    c. Do you need to know whether *myList* is an array list or linked list orientation in order to insert an element? Why or why not?

    Circle: Yes No

Briefly explain your choice:_____

_____