

CSE 123 Autumn 2025 Practice Quiz 1

Name of Student: _____

Section (e.g., AA): _____ Student Number (e.g. 1234567) : _____

Do not turn the page until you are instructed to do so.

Rules/Guidelines:

- You must not begin working before time begins, and you must stop working **promptly** when time is called. Any modifications to your quiz (writing or erasing) before time begins or after time is called will be reported as academic misconduct to the university.
- You are allowed one page of notes, no larger than 8.5 x 11 inches. You may not access any other resources or use any electronic devices (including calculators, phones, or smart watches, among others) during the quiz. Using unauthorized resources or devices will be reported as academic misconduct to the university.
- **We will only scan your exam packet.** We will not scan your scratch paper or your notes sheet. We have included a blank page in the exam packet. If you run out of space while answering a question, write your answer on that blank page and clearly indicate that we should look for your answer there.
- In general, you are limited to Java concepts or syntax covered in class. You may not use `break`, `continue`, a `return` from a `void` method, `try/catch`, or Java 8 stream/functional features.
- You are limited to the standard Java classes and methods listed on the provided reference sheet. You do not need to write import statements.
- If you abandon one answer and write another, **clearly cross out** the answer(s) you do not want graded and **draw a circle or box** around the answer you do want graded. When in doubt, we will grade the answer that appears in the space indicated, and the first such answer if there is more than one.
- If you require scratch paper, raise your hand and we will bring some to you.
- Answers must be written as proper Java code. Pseudocode or comments will not be graded.
- The quiz is not graded on code quality. You are not required to include comments.
- You are also allowed to abbreviate `System.out.print` and `System.out.println` as `S.o.p` and `S.o.pln` respectively. You may **NOT** use any other abbreviations.

Grading:

- There are three problems. Each problem will receive a single E/S/N grade.
- Minor syntax errors will be ignored as long as it is unambiguous what was intended (e.g. forgetting a semicolon, misspelling a variable name where there is only one close option). Major syntax errors, or errors where it is unclear what was intended, may have an impact on your grade.

Advice:

- Read all questions carefully. Be sure you understand the question *before* you begin your answer.
- The questions are not necessarily in order of difficulty. Be sure you at least attempt every question.
- Write clearly and legibly. We cannot award credit for answers we cannot read.
- If you have questions, raise your hand to ask. The worst that can happen is we will say "I can't answer that."
- Ask questions as soon as you have them. Do not wait until you have several questions.

Initial here to indicate you have read and agreed to these rules:

1. Runtime

For each row below, analyze the operation by marking an "X" in the box for the appropriate complexity class.

Unless otherwise stated:

- assume `LinkedList` has only a `front` field (and no other fields, such as size or back)
- n means the length of the list before the operation

Operation	$O(1)$	$O(n)$	$O(n^2)$
<code>ArrayList.get(index)</code>			
<code>LinkedList.get(index)</code>			
Adding an element to the end of an <code>ArrayList</code>			
Adding an element to the end of a <code>LinkedList</code>			
<code>ArrayList.add(index, value)</code>			
<code>LinkedList.add(index, value)</code>			
<code>ArrayList.size()</code>			
<code>LinkedList.size()</code>			
Calling this method: <pre>public static void m(int[] data) { for (int i = 0; i < data.length; i++) { System.out.println(data[i]); } } where n is data.length</pre>			
Calling this method: <pre>public static void m(int[] data) { for (int i = 0; i < 100000; i++) { System.out.println(data[i]); } } where n is data.length</pre>			
Calling this method: <pre>public static void m(int[] data) { for (int i = 0; i < data.length * data.length; i++) { System.out.println(data[i % data.length]); System.out.println(data[i / data.length]); } } where n is data.length</pre>			

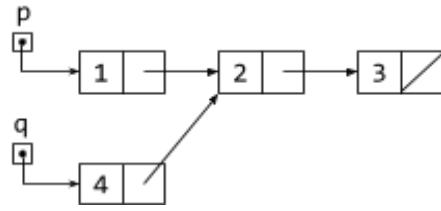
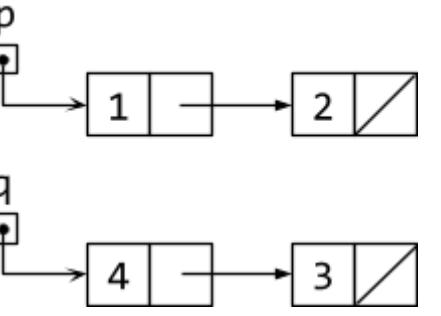
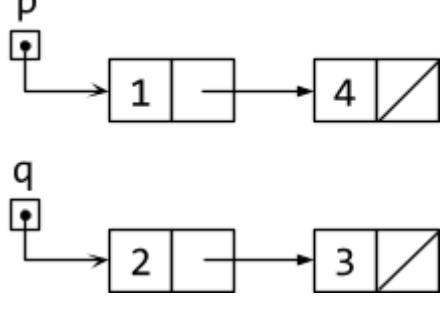
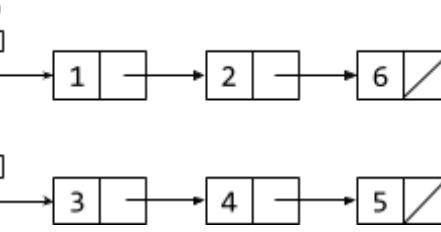
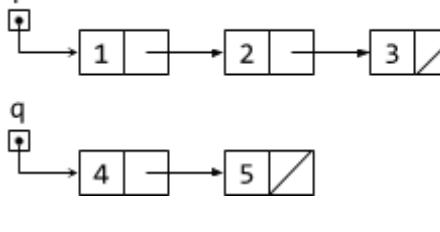
2. Analysis

Part A: ListNode Manipulation

In the following table, the “Before” column shows a diagram of some linked nodes, the “Code” column specifies some code to be applied to the nodes in the before diagram, and the “After” column shows a diagram of the nodes after the code has been applied.

Complete the table, filling in either the before picture, the code, or the after picture. You should not create any new `ListNode`s or modify any `.data` fields, and **there should be only one instance of each node with a specific value**. The after picture does not need to show any temporary references that were created.

Your `ListNode` diagram format doesn't have to match that of the problem so long as it is clear what you intend. In your code, you may use as many temporary references as you'd like to accomplish your goal, but you may *not* create any new `ListNode` objects. You also may not change the data field of any `ListNode` objects

Before	Code	After
	<pre>q.next = p.next.next; ListNode temp = p.next; p.next = null;</pre>	
		
		

Part B: Data Structure Design

Consider each of the following data structures which could be implemented using a list. In each case, consider only the operations listed when making your determination. . Decide which of these statements is true for each given situation:

- A. An ArrayList would implement this more efficiently.
- B. A LinkedList would implement this more efficiently.
- C. ArrayList and LinkedList would implement this equally efficiently.

Where one implementation is considered “more efficient” than the other if its running time is a smaller complexity class. They are equally efficient if their running times are the same complexity class.

In all cases, you may ignore the possibility of resizing the underlying array for an ArrayList.

Data structure	Operations	True statement		
A stack where the top of the stack is the first element in the underlying list	push(int val) pop()	A	B	C
A queue where the front of the queue is the first element in the underlying list	add(int val) remove()	A	B	C
A set (does not store duplicate values)	add(int val) contains(int val)	A	B	C
A list when items will always be added to the end	add(int val) remove(int index) get(int index)	A	B	C

3. Implementation

Implement a method called `removeAll(Set<Integer> targets)` within the `LinkedList` class that takes in a Set of Integers to remove. If any `ListNode` has a value that exists inside the given Set, that node will be removed from the chain.

For example, suppose a `Set` `nums` contains the values `{1, 4, 9}` and the contents of a `LinkedList` `list` is `[9, 4, 7, 5, 4, 2, 1]`. Then, after the call `list.removeAll(nums)`, `list` will contain `[7, 5, 2]`.

The code for `LinkedList` is on the reference sheet provided. Your solution may not call any existing methods of the `LinkedList` class.

CSE 123 Quiz 1 Reference Sheet

(DO NOT WRITE ANY WORK YOU WANTED GRADED ON THIS REFERENCE SHEET. IT WILL NOT BE GRADED)

Math Methods

<code>abs(x)</code>	Returns the absolute value of <code>x</code>
<code>max(x, y)</code>	Returns the larger of <code>x</code> and <code>y</code>
<code>min(x, y)</code>	Returns the smaller of <code>x</code> and <code>y</code>
<code>pow(x, y)</code>	Returns the value of <code>x</code> to the <code>y</code> power
<code>random()</code>	Returns a random number between <code>0.0</code> and <code>1.0</code>
<code>round(x)</code>	Returns <code>x</code> rounded to the nearest integer

String Methods

<code>charAt(i)</code>	Returns the character in this String at a given index
<code>contains(str)</code>	Returns <code>true</code> if this String contains the other's characters inside it
<code>endsWith(str)</code>	Returns <code>true</code> if this String ends with the other's characters
<code>equals(str)</code>	Returns <code>true</code> if this String is the same as <code>str</code>
<code>equalsIgnoreCase(str)</code>	Returns <code>true</code> if this String is the same as <code>str</code> , ignoring capitalization
<code>indexOf(str)</code>	Returns the first index in this String where <code>str</code> begins (-1 if not found)
<code>lastIndexOf(str)</code>	Returns the last index in this String where <code>str</code> begins (-1 if not found)
<code>length()</code>	Returns the number of characters in this String
<code>isEmpty()</code>	Returns <code>true</code> if this String is the empty string
<code>startsWith(str)</code>	Returns <code>true</code> if this String begins with the other's characters
<code>substring(i, j)</code>	Returns the characters in this String from index <code>i</code> (inclusive) to <code>j</code> (exclusive)
<code>substring(i)</code>	Returns the characters in this String from index <code>i</code> (inclusive) to the end
<code>toLowerCase()</code>	Returns a new String with all this String's letters changed to lowercase
<code>toUpperCase()</code>	Returns a new String with all this String's letters changed to uppercase
<code>compareTo(str)</code>	Returns a negative number if this comes lexicographically (alphabetically) before other, 0 if they're the same, positive if this comes lexicographically after other.

LinkedList Class

```
public class LinkedList extends AbstractIntList {
    private ListNode front;

    public LinkedList() {...}
    public LinkedList(int[] nums) {...}
    public void add(int index, int value) {...}
    public int remove(int targetIndex) {...}
    public int size() {...}
    public int get(int index) {...}

    public static class ListNode {
        public final int data;
        public ListNode next;

        public ListNode(int data) {...}
        public ListNode(int data, ListNode next) {...}
    }
}
```