

Name: _____

Question 1. Selecting Implementations (10 points)

For each of the following scenarios, first **circle** the solution that you would pick and then **explain why** below.

Please answer five out of the six questions below.

Example:

Keeping track of when students get to ask questions to a TAs during Tutoring Hours

i. Stack

ii. Queue

iii. Dequeue

Justification: Students should get to ask their questions in the order that they've been waiting, with students who have been waiting longer getting an opportunity first. FIFO is best implemented using a queue.

(A) Looking for a single element within an unordered collection

i. Linear search

ii. Binary search

Justification: _____

(B) Using binary search to identify an element within an ordered collection

i. Array

ii. Linked List

iii. Doesn't matter

Justification: _____

(C) Storing a set of items sold in the Campus Center and their associated prices.

i. Hash Set

ii. Lookup Table

iii. Hash Map

Justification: _____

Name: _____

(D) Sort a massive collection of items, where only a small fraction of the items can be held in memory at the same time

i. Selection Sort

ii. Merge Sort

Justification: _____

(E) Track friendships and friend-breakups among the first-year class

i. Array

ii. Tree

iii. Graph

Justification: _____

(F) Organize data that you need search for several different things

i. Array

ii. Linked List

iii. Binary Search Tree

Justification: _____

Reminder: You can skip one of the above problems. If you answer all 6, please indicate which one you would like to be excluded.

Name: _____

Question 2. (Flawed) Operations on Singly Linked Lists (12 points)

Consider the list methods implemented below. Each one has at least one bug. This can be either a general issue with the implementation or a special case that is not handled properly. It will not be a syntax error.

For each method below, you are asked to indicate what problem might arise from the implementation shown. Please see the example below.

Please answer four of the five questions below.

Example:

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

What's wrong? This method will create two new nodes. The previous head node is no longer part of the list. Instead, its data has been copied over into a new node.

(A) What else could go wrong with the method in the example above (think of special cases)?

Name: _____

(B)

```
65  /** Returns the last element in the linked list */
66  public NodeSL<T> getTail(){
67      NodeSL<T> current = head;
68      while(current.next.next != null){
69          current = current.next;
70      }
71      return current;
72  }
```

What is wrong with this method?

(C)

```
74  /** Calculate the size of the list by counting the nodes */
75  public int size(){
76      int size = 0;
77      if(!isEmpty()) {
78          NodeSL<T> node = this.head;
79          while (!(node.getNext() == null)){
80              size++;
81          }
82      }
83      return size;
84  }
```

What is wrong with this method?

Name: _____

(D)

```
162  /**
163     * Removes the given item from the head of the list
164     * @return v item removed
165     */
166  public T removeFirst() {
167     T result = head.getData();
168     head = head.getNext();
169     return result;
170 }
```

What is wrong with this method?

(E)

```
22  /** Copy constructor makes a deep copy of the list */
23  public SLL(SLL<T> orig) {
24     if (orig.head == null) {
25         // handle special case of an empty list
26         head = null;
27     } else {
28         head = orig.head;
29     }
30 }
```

What is wrong with this method?

Reminder: You can skip one of the above problems. If you answer all 5, please indicate which one you would like to be excluded.

Name: _____

Question 3. Iterative and Recursive Problem Solving (23 points)

This semester, we talked about both iterative and recursive strategies for solving problems. Imagine you have an array of integers of length n and you want to find the sum. Assume n is available as `array1.size`.

Here is an example array with $n=7$.

2	4	0	6	1	8	3
---	---	---	---	---	---	---

- A. How would you take the sum of this array iteratively? Please provide pseudocode describing your approach.

Name: _____

- B. With an **iterative** approach, what is the computational complexity of taking the sum of all the elements in an **array**? Feel free to show your work, but please put your final answer in the brackets.

$O(\quad)$

The next few questions will ask you to consider how to use recursion to take the sum of an array

- C. With each recursive call, how would the data you are looking at (e.g., the passed argument) change? It may help to think about whether this is a structural or generative recursive problem. You can answer in words or draw a picture.

- D. Based on your answer to D, what would be the base-case?

Name: _____

- E. In pseudocode, sketch out a recursive function to calculate the sum of an array of length n . If you can't think of how to write it in pseudocode, just write out your logic and/or describe how you would approach the problem.

Name: _____

Question 4. Computational Complexity of Linked-List Operations (12 points)

Imagine you have a linked list containing integers as the data.

Outline **two** approaches that would allow you to find the ***k* lowest elements**. Indicate the **computational complexity** of each approach. Explain your approach using pseudocode, words, and/or drawings. Please provide a short name/title for each approach so that I know approach you are employing (e.g., "Calculate the sum while counting the length, divide sum/length" would be a way to find the average.)

Approach 1: _____

How it works:

Computational complexity: _____

Name: _____

Approach 2: _____

How it works:

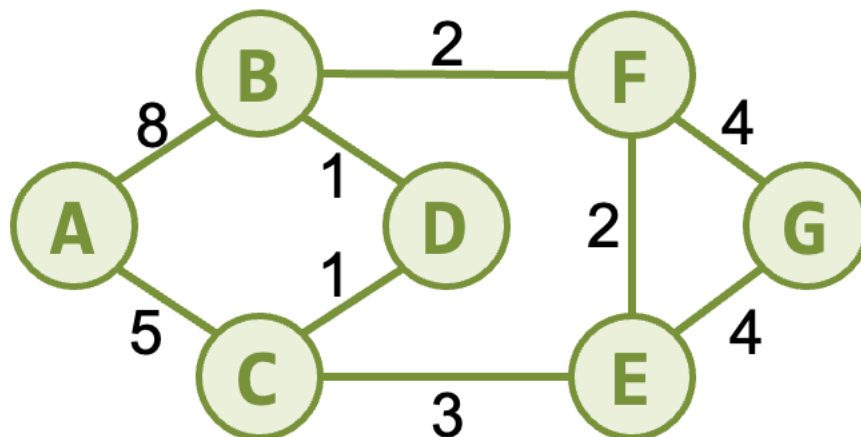
Computational complexity: _____

Name: _____

Question 5. Dissecting an Algorithm (28 points)

Dijkstra's algorithm gives us a way to find the shortest path between two nodes. We did not discuss it this semester, so **you are not expected to have any prior knowledge** of this algorithm. Instead, your goal is to figure out how you would implement Dijkstra's algorithm using what you've learned this semester.

We will consider a graph with weighted edges, like this:



Your goal is to find the cheapest path (in terms of the sum of the edges visited) to get from **Node A to Node G**. To do this, you will start at node A. When you **visit** Node A, you identify which nodes you can reach from A and what the associated cost of reaching them is.

There are a few things you need to keep track of, as described in **parts A-C** below. For each, propose a data structure that can be used for this goal, and explain why.

PART A: First, you want to **keep track of nodes that have not yet been visited**. Initially, you want track all of the nodes in your graph, as none of them have been visited yet. As you visit each node, you remove it from your Unvisited collection. The Unvisited nodes do not need to be stored in any particular order. Instead, you want to be able to find and remove them quickly.

- i. The most efficient data structure for tracking Unvisited nodes is:

Name: _____

ii. Why?

PART B: Next, you'll need to track the **cost** of the most efficient route you've found to get to each node from A. We would want to organize this information so that we can easily look up the lowest cost we have found so far.

Example: In the graph above:

1. You can get from $A \rightarrow B$ with a cost of 8.
2. You can get from $A \rightarrow C$ with a cost of 5.
3. Next, you find you can get from $A \rightarrow D$ with a cost of 9 (8 to get to B and 1 to get from $B \rightarrow D$).
4. Later, you discover that you can get from $A \rightarrow D$ with a cost of 6 (5 + 1 additional from $C \rightarrow D$).
 - a. Because 6 is less than 9, we would want to remove the value of 8 and instead store the cost of 6 to reach D from A.

Think about ways to store and access this information.

- i. The most efficient data structure for tracking the minimum cost to reach each node is: _____
- ii. Why did you choose this data structure?

Name: _____

- iii. What would you want to set the costs to initially, before you start visiting the graph? (Please explain why.)

PART C: We need to track not only the minimum cost we've found to reach a node (as in Part B) but also the path we took to get there. In particular, we want to know which neighbor we came from to get the lowest cost. This will allow us to retrace our steps later.

Example:

Continuing the example above:

1. We visited A to get to B.
2. We visited A to get to C.
3. We visited B to get to D.
4. Now we found we can get to D via a cheaper route, so we want to replace B with C.

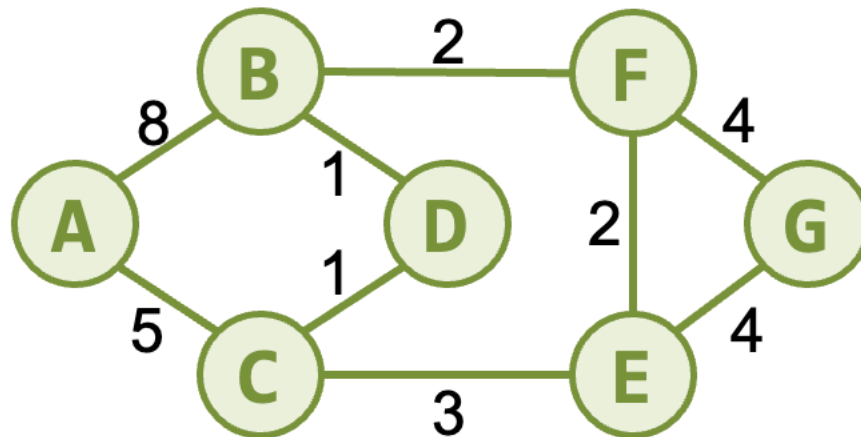
- i. The most efficient data structure for tracking the neighbor that led to the lowest-cost route to each node is:

- ii. Why?

Name: _____

PART D: What is the lowest-cost path you can find from A to G? Please show all of your work. (You are not expected to execute this perfectly on the first time, so show your logic clearly—there will be lots of partial credit given). You can traverse the nodes however you like. The main goal is to show how you would apply your answers to A-C to solving this problem.

If you feel stuck: Use your answers and the information provided above to show how you would find the lowest-cost path from A → D. Then write down the questions you have about how to continue searching for G.



Lowest Cost Found from A → G:

Name: _____

(Additional space for Problem 6)

END OF QUESTIONS

Name: _____

EXTRA PAGE FOR SCRATCH WORK:

Name: _____

EXTRA PAGE FOR SCRATCH WORK: