CptS 122 Spring 2025 Final Review

Question 1

Examine the following code snippet and identify any errors present:

```
class A {
   int foo() const {
      return x = 10;
   }

   int bar() {
      return x = 49;
   }

   void reset() {
      x = -1;
   }

   int x;
};
```

Indicate the issues you found in the code and provide suggestions for how to fix them.

Question 2

What is the worst-case time complexity in Big-O notation for inserting an element into a Binary Search Tree (BST)?

Question 3

What is the worst-case time complexity in Big-O notation for the **enqueue** operation in a Queue?

*Assume the Queue is implemented using pointers: Node * mpHead and Node *mpTail for the Queue.*

Question 4

What is the worst-case time complexity in Big-O notation for the **dequeue** operation in a Queue?

*Assume the Queue is implemented using pointers: Node * mpHead and Node *mpTail for the Queue.*

Question 5

Determine the Big-O time complexity for the following functions, assuming a singly-linked list is used as the underlying data structure:

1. BST::search(for: T)
2. List::insertAt(index: Int)
3. Stack::push(data: T)
4. Stack::pop(store: T&)
5. Stack::isEmpty()
6. BST::delete(item: T)
7. List::insertAtEnd(item: T) (Assume we have pHead and pTail)
8. List::insertAtEnd(item: T) (Assume we have pHead)

9. List::removeFromFront(item: T) (Implemented using a C-style array)

Question 6

Consider the following operations on a List. For each operation, identify the best underlying data structure. Write down any assumptions you have made.

a) **Read** an item at position n

b) **Insert** an item at position n

c) **Remove** an item from the front

d) **Add** an item at the end

We have the following options:

1. Singly linked nodes
2. Doubly linked nodes
3. Primitive array
4. std::vector

Question 7

Examine the following pseudocode and describe its functionality. Check your answer by writing C++ code.

```
Init stack with 200 numbers
Init empty list
While ( the stack is not empty ) do
        Pop element from the stack and put at the FRONT of the list.
Done
While ( the list is not empty) do
        Remove element from the END of the list and push to the stack
Done
```

Question 8

Examine the following pseudocode and describe its functionality. Check your answer by writing C++ code.

```
Init stack with 200 numbers
Init empty list
While ( the stack is not empty ) do
        Pop element from the stack and put at the FRONT of the list.
Done
While ( the list is not empty) do
        Remove element from the FRONT of the list and push to the stack
Done
```

Question 9

What pointer(s) **will** be modified when the Queue::dequeue() is called on a **non-empty** queue?

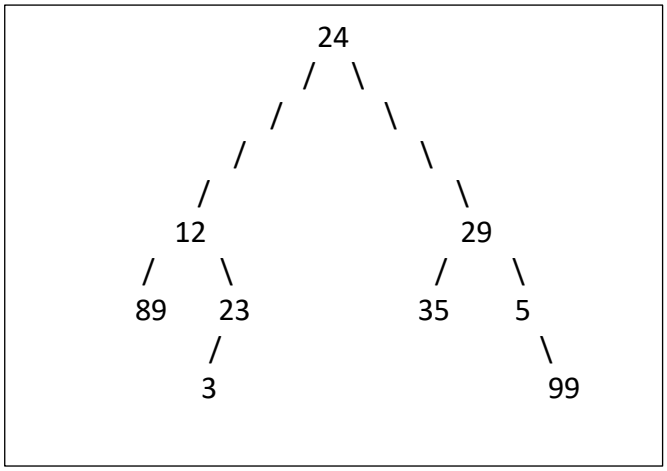
1. Tail
2. Head
3. None
4. Head & Tail
5. None of the above (*Please describe what you believe the answer is*)

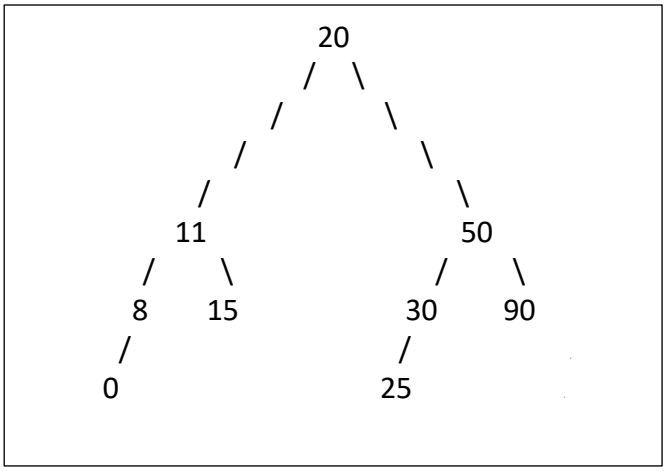
Question 10

What is the difference between a binary tree and a binary search tree?

Question 11

For the given diagrams, identify the pre-order, post-order, and in-order traversals. Additionally, specify the names of these trees.





Question 12

Briefly describe each of the following traversals for a Binary Search Tree (BST). For each traversal, please provide a use case:

1. Pre-order
2. Post-order
3. In-order

Question 13

Explain how memory is allocated for pointers and arrays in C++. Additionally, describe the process for deallocating memory in C++.

Question 14

Complete the following protection mode matrix. The rows represent the inheritance mode in the derived class, while the columns represent the declared member's protection level in the base class.

Protection Mode	public	protected	private
public			
protected			
private			

Question 15

From the following options, identify the "Big Three" components in C++:

1. Constructor
2. Virtual destructor
3. Copy Constructor
4. Getters/setters
5. Destructor
6. Default Constructor
7. Virtual functions
8. Overloaded assignment operator
9. Encapsulation

Question 16

What are some reasons a data structure might provide two different implementations of operations such as insert, enqueue, search, etc.? Please explain your reasoning.

Question 17

What is the term used to describe a class that contains one or more pure virtual functions?

Question 18

What is the purpose of the keyword "virtual" in C++?

Question 19

What are the different types of inheritance in C++? Provide a brief description of each type.

Question 20

Define the terms "upcasting" and "downcasting" in the context of object-oriented programming.

Question 21 (a)

We want to create a Parser class with very basic functionality. The constructor will accept a string. There will be a parse function which accepts a vector of strings. We can only parse up to 1000 characters and generate up to 500 tokens.

Declare the following members of Parser.

1. A constructor which accepts a reference to a constant string.
2. A function called parse which accepts a std::vector<string> by reference to place tokens.
3. A private data member which holds the string to parse.

Question 21 (b)

Define the constructor for Parser. We have defined an exception, InputLengthExceededException, which inherits from std::runtime_exception. If the input is too long, throw an exception and provide a detailed error message.

Question 21 (c)

Define the parse function. We have created an exception called InputLengthExceededException, which derives from std::runtime_exception. If an excessive number of tokens are produced, throw an InputLengthExceededException with a detailed message. Assume that the input string is in CSV format and does not contain any commas within the values. *Write down assumptions*.

Hint: use s1.substr(i, c) and s1.find("str")
S1.substr(i,c) will return a string with `c` characters from index i.
S1.find("str") will return an index where the substring "str" was found. If "str" is not found, std::string::npos is returned.

Example:

```
Int index = S1.find(",")

S1 = "Hello, World!"

Hi = s1.substr(0,index);

World = s1.substr(index);

Cout << Hi << endl; // output: `Hello`

Cout << World << endl; // output: `, World!`
```

Solution to be updated soon to GitHub - Final Review - SwiftlyDesigner. Guaranteed by Friday 23:59