Data Structures and Fundamentals of Programming

Problem 1
In C++ implement a **generic** class, called `Queue<T>`, that uses a **single-linked list** implementation. This should implement the **queue** abstract data type (ADT). It must be generic on the type of the data to be stored. Give all class definitions and implement the following for `Queue`:

- Default constructor
- Destructor
- Copy-constructor
- Swap that runs in constant time no matter what the length of the queues
- Assignment operator – using standard C++ copy semantics
- `enqueue(T)` – takes a parameter of type T and adds it to the queue
- `T dequeue()` – removes an item from the queue

Your implementation can **NOT** use STL or any other libraries (standard or otherwise).

Problem 2
In C++, implement a String abstract data type (ADT) using a dynamically allocated `char` array. The array of `char` must be `NULL` terminating. This dynamic version of the String will only allocate **exactly** the amount of memory necessary to store the characters. That is, the length will **always** be the same as the capacity. However, the size of the dynamic array needs to have an extra space for the `NULL` terminator. You must implement the following methods:

- Default constructor that sets the object to the empty String.
- Constructor that takes a `const char` array and converts it into a String.
- Copy constructor
- Destructor
- Swap – swaps two strings in constant time regardless of the size of the array.
- Assignment operator using standard C++ copy semantics
- `String operator+(const String&) const;` that concatenates any two Strings and returns a new String with the proper amount of allocated memory.

Your implementation can **NOT** use STL or any other libraries (standard or otherwise). You **cannot** use `std::string`.

Problem 3
In C++ implement a **ternary tree** abstract data type (ADT) that uses **dynamic memory allocation**. Make it a tree of integers. Each node will have between 0 and 3 children (left, middle, right). Along with the class definition(s), you must implement the following methods for the class `ternary`:

- Default constructor
- Destructor – **must** be recursive or use a recursive method to delete all the nodes in a tree.
- Copy-constructor – **must** be recursive or use a recursive method make an complete copy of a tree.
- Preorder – which prints out the entire tree using a preorder traversal. **Must** be recursive.
- Postorder – which prints out the entire tree using a postorder traversal. **Must** be recursive.

Your implementation can **NOT** use STL or any other libraries (standard or otherwise).