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 will implement the **queue** abstract data type (ADT). It will 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 regardless of the length of the queues
- Assignment operator – using standard 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 array. The array of char is to be NULL terminating. This dynamic version of the String only allocates **exactly** the amount of memory necessary to store the characters and NULL terminator. That is, the length will **always** be the same as the capacity. However, the size of the dynamic array needs to have an extra char 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 strings.
- Assignment operator using standard copy semantics
- Concatenation (`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 **binary search tree** abstract data type (ADT) that uses **dynamic memory allocation**. Make it a tree of integers. Along with the class definition(s), you must implement the following methods for the class:

- Default constructor
- Destructor – **must** be recursive or use a recursive method to delete the nodes.
- Copy-constructor – **must** be recursive or use a recursive method to copy the nodes.
- `insert` which takes a parameter of type integer and creates a new node that is added to the tree in the correct position based on the rules of a binary search tree. This **must** be recursive.

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