Data Structures and Fundamentals of Programming

Problem 1

In C++ implement a generic **double**-linked-list class, called `List<T>`, that uses dynamic memory allocation. The list must look like the following:

`frontptr -> X₁ <->X₂ <-> … <-> Xₙ <- backptr`

where X_1 is the first node in the list and X_n is the last node in the list. Besides `List`, you will need a class called `node<T>`. Along with the class definition(s) you will need to implement a following member functions for `List<T>`:

- Default constructor
- Copy constructor
- Destructor
- `addToBack()` – Adds an item to the back of the list.
- `addToFront()` – Adds an item to the front of the list.
- `T remove(node<T>*)` – removes a node from the list, given a pointer to the node.

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

Problem 2

In C++ implement a **generic** class, called `Queue<T>`, that uses a **fixed size circular array** implementation. This should implement the queue ADT. It should be generic on the type of the data to be stored. The implementation must be able to utilize the entire array in storing items. Give all class definitions and implement the following for `Queue`:

- Default constructor
- `push(T)` – takes an parameter of type T and adds it to the queue
- `T pop()` – removes a item from the queue
- `isEmpty()` – returns true when the queue is empty.
- `isFull()` – returns true when the queue is full.

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

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.

Also answer the following questions:

- What tree traversal algorithm is used for the destructor?
- What tree traversal algorithm is used for the copy constructor?

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