Problem 1
In C++, implement the abstract data structure (ADT) queue as a generic class, called `Queue<T>`. Implement the queue using a circular array, it must use a fixed sized array. It will be generic on the type of the data to be stored. The implementation must utilize the entire fixed sized array in storing items. Give all class definitions and implement the following for `Queue`:

- Default constructor
- `push(T)` – takes a 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.

You can NOT use STL or any other predefined library or built in types (such as std::string).

Problem 2
In C++ implement a list/iterator using a generic double-linked-list that uses dynamic memory allocation. The list must look like the following:

```
beginning --> X₀ <-> X₁ <-> ... <-> Xₙ <-- ending
```

where X₀ is the first node in the list and Xₙ is the last node in the list. Besides the class called `List`, you will need class called `dnode`. Along with the class definition(s), you must implement the following methods, using standard semantics, for `List<T>`:

- `List()` - Default constructor
- `~List()` - Destructor
- `List(const List<T>&)` - Copy-constructor
- `insertAfter(const T& val, Itr<T> it)` – Adds an item after the node pointed to by the iterator.
- `remove(Itr<T> it)` – removes the node pointed to by the iterator.
- `begin()` – returns an iterator to the front of the list
- `end()` – returns an iterator to the back of the list

Implement an iterator class called `Itr` for the `List` class. For the iterator you must write the following methods with standard semantics/behavior:

- `Itr()` - Default constructor
- `Itr(dnode<T>* ptr)` - constructor
- `operator++()` - pre-increment with standard semantics
- `operator++(int)` - post-increment with standard semantics

Note: 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.

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