Data Structures and Fundamentals of Programming

Problem #1

In C++ implement a **generic** class, called `Stack<T>`, that uses a **single-linked list** implementation. It implements the **stack** 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 `Stack`:

- Default constructor
- Destructor
- Copy-constructor
- Swap that runs in constant time no matter how many items are on the stacks
- Assignment operator – using standard copy semantics
- `push(T)` – takes a parameter of type T and adds it to the top of the stack
- `T pop()` – removes an item from the top of the stack

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

Problem #2

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 -> X_1 <->X_2 <-> … <-> X_n <- 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 or struct 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
- `insertBefore(const T&, node<T>*)` – Adds an item in list before pointer to any node in list
- `insertAfter(const T&, node<T>*)` – Adds an item in list after pointer to any node in list
- `T Remove(node<T>*)` – removes a node from the list, given a pointer to the node

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

Problem #3

In C++ implement a **generic** class, called `darray<T>`, that implements a **dynamic array** of any type. The array needs to be resizable either larger or smaller. Must use new and delete to allocate the array. You must implement the following methods:

- Default constructor – creates a zero sized array
- Constructor that takes an integer and creates an array of that size
- Copy constructor
- Destructor
- Swap – swaps two darray in constant time regardless of the size of the arrays
- Assignment operator using standard copy semantics
- Resize – make the array larger/smaller given a new size (integer). Must preserve contents of array being resized (to the extent possible)

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