Task 1
*Define and implement a class **DoubleNode** that is capable of holding an item of any arbitrary type **ItemType**.
*As a node of a Doubly-Linked list, it should contain two pointers that respectively point to other objects of type **DoubleNode**.

The following methods will be required of your DoubleNode class, but feel free to add methods as you see fit:

Default Constructor

Parameterized Constructor(s)

ItemType getItem( ) const

bool setNext(DoubleNode* ptr)

bool setPrev(DoubleNode* ptr)

*Entitle your header file **DoubleNode.hpp**, and entitle your implementation file **DoubleNode.cpp***

Task 2
*Define and implement a class **DoublyLinkedList** that is a demonstration of the Doubly-Linked List concept.
*It should contain a head pointer to a DoubleNode of any arbitrary type **ItemType**, and it should contain a member that keeps track of its size.
*Let **DoublyLinkedList** be 1 indexed unlike arrays, which are 0 indexed.

The following methods are required of your **DoublyLinkedList** class:

Default Constructor *//create a default constructor*

Copy Constructor *//create a deep copy constructor*

Destructor *//create a destructor*

bool insert(const ItemType &item, const int &position) *//inserts item at position in caller list*

bool remove(const int &position) *//removes the node at position*

int getSize( ) const *//returns the number of the nodes in the calling list*

DoubleNode<ItemType> *getHeadPtr() const *//returns a copy of the headPointer*

DoubleNode<ItemType> *getAtPos(const int &position) const *//returns a pointer to the node located at position*

bool isEmpty( ) const *//returns whether the calling list is empty*

void clear( ) *//clears the list*

void display( ) const *//prints the contents of the calling list in order*

void displayBackwards( ) const *//prints the contents of the calling list in reverse order*

DoublyLinkedList<ItemType>interleave(const DoublyLinkedList<ItemType> &a_list)} *//returns the interleaved list of the calling and parameter lists*

*Interleave Example: Define the calling list as a set of ordered nodes, ***L1 = {4, 2, 8, 5, 8}***, and define the list that is passed as a parameter as a set of ordered nodes, ***L2 = {5, 1, 8, 4, 5, 9}***.

L1.interleave(L2) gives the set {***4, 5, 2, 1, 8, 8, 5, 4, 8, 5, 9}***

*In other words, to create the interleaved list, first add a node from L1, then one from L2, and then repeat as many times as necessary.
*If there are any nodes left over in L1 or L2, add them to the end of the list.

*Entitle your header file **DoublyLinkedList.hpp**, and entitle your implementation file **DoublyLinkedList.cpp**.*

Required Files: ***DoubleNode**.hpp **DoubleNode**.cpp **DoublyLinkedList**.cpp **DoublyLinkedList**.hpp*