================================

Node.h

===============================

class Node

{

public:

Node(T);//set nextPtr to null and initialize the data with the value passed to constructor

T getData()const;//return the value stored in the data variable

Node<T>\* getNextPtr()const;//return the address stored in nextPtr variable

void setData(T);//set the passed value to the data variable

void setNextPtr(Node\*);//set the passed value to the nextPtr variable

private:

T data;

Node<T>\* nextPtr;

};

=================================

Node.cpp

=================================

using namespace std;

#include<iostream>

#include "node.h"

template<class T>

Node<T> :: Node (T a)

{

data = a;

nextPtr = NULL;

}

template<class T>

void Node<T> :: setData(T a)

{

data = a;

}

template<class T>

void Node<T> :: setNextPtr (Node<T> \*p)

{

nextPtr = p;

}

template<class T>

T Node<T> :: getData() const

{

return data;

}

template<class T>

Node<T> \* Node<T> :: getNextPtr() const

{

return nextPtr;

}

====================================

Queue. h

=====================================

#include<iostream>

using namespace std;

#include "node.h"

template<class T>

class Queue

{

private:

Node<T> \*head;

public:

Queue();//set head to null

~Queue();// ???

void enqueue(T);//add a new node at the end of linked list with data provided as argument

T dequeue();//return the data stored in first node and delete the first node

T peek();//return the data stored in first node

bool isEmpty();//return true if there is no node in Queue

};

==============================

Queue.cpp

==============================

using namespace std;

#include<iostream>

#include "queue.h"

template<class T>

Queue<T> :: Queue()

{

head = NULL;

}

template<class T>

Queue<T> ::~Queue()

{

Node<T>\* current = head;

while(current!= NULL)

{

Node<T> \*temp = current;

current = current->getNextPtr();

delete temp;

}

}

template<class T>

void Queue<T> :: enqueue(T a)

{

Node<T> \*temp = new Node<T>(a);

if(head == NULL)

{

head = temp;

}

else

{

Node<T>\* current = head;

while(current->getNextPtr() != NULL)

{

current = current->getNextPtr();

}

current->setNextPtr(temp);

}

}

template<class T>

T Queue<T> :: dequeue()

{

if(head != NULL)

{

T temp\_data = head->getData();

Node<T> \*temp = head;

head = head->getNextPtr();

delete temp;

return temp\_data;

}

}

template<class T>

T Queue<T> :: peek()

{

if(head != NULL)

{

return head->getData();

}

}

template<class T>

bool Queue<T> :: isEmpty()

{

if(head != NULL)

{

return false;

}

else

{

return true;

}

}