Homework#5 Due Date: 4/10/17 @ 11:55pm

Pages 156 exercises

#2 (10 pts) - Revise the add method of the LinkedBag class so that the new node is inserted at the end of the linked chain instead of the beginning. Pseudocode is ok but if possible try to write C++ code (this is to help you prepare for the midterm).

template<class ItemType>

bool LinkedBag<ItemType>::add(const ItemType& newEntry)

{

//new node is inserted @ end of linked chain

// headPtr will always be in the front of the linked chain

// therefore we need to iterative through the chain to get to the end

// once we reach the end we will set a pointer to point to the newNode

// Add to beginning of chain: new node references rest of chain;

// (headPtr is null if chain is empty)

Node<ItemType>\* newNodePtr = new Node<ItemType>();

newNodePtr->setItem(newEntry);

newNodePtr->setNext(nullptr); //end will always be a nullptr

if(itemCount == 0) {

headPtr = newNodePtr;

} else {

Node<ItemType>\* last = headPtr; //iterate through the list starting from the beginning

for(int iterate = 1; iterate < itemCount; ++iterate) {

last = last->getNext(); //get to the end point

}

last->setNext(newNodePtr); //set pointer of lastNode to newNodePtr

}

itemCount++;

return true;

} // end add

|  |
| --- |
|  |

#3 (10 pts) - revise the getCurrentSize member function of LinkedBag so that it would work if the LinkedBag didn't have the member variable itemCount. You only have to do part a - the iterative solution. Pseudocode is ok on this one but again try the C++ if possible.

int LinkedBag<ItemType>::getCurrentSize() const

{

//revise getCurrentSize so it counts the Nodes in the chain...

int counter = 0;

Node<ItemType>\* current = headPtr;

while(current != nullptr) {

counter++;

current = current->getNext();

}

return counter;

}

|  |
| --- |
|  |

#10 (10 pts) - note you only need to define the class for the Node (not for the list) - just the .h file is fine. This doesn't have to be perfect just give good ideas on how you would structure the Node class for a doubly linked list.

|  |
| --- |
|  |