**Homework Prog 260 – Write 2 new Linked List Methods**

Modify the supplied project by completing the code for the last 2 methods in the IntLinkedList class. 2 new empty methods are already there waiting for your code. Here are the 2 methods:

// Method to add a node to the end of the list:

public void InsertAtEnd(int value)

{

}

// Method to remove a node from the end of the list which returns an int with the value that was in that node

public int RemoveFromEnd()

{

}

Notice that Program.cs has a code to test these functions. When you have your code correct, that program will write out the values that match the comments in it. It should look like this but just for a split second, then it will throw and exception because the code tries to remove a 4th element.



That’s all you need to know, but I will make a few suggestions.

For InsertAtEnd

* Check if the list is empty, if it is, you can just make a call to InsertAtFront and you are done!
* If the list is not empty, create a new node, and set its value equal to the passed in value from the method call. Then build a While loop that walks down the linked list until it finds the last node. When it finds the last node, it exits the while loop, and adds the new node at the end. **It only takes one line of code to add the node to the end.** And For this method, you really can walk to the last node, you do not have to stop one node before the last node (as you will have to in the next method)

For RemoveFromEnd

* Check if the list is empty, if it is, throw an exception

throw new IndexOutOfRangeException("Linked List is empty");

* Otherwise, check if there is only one node in the list, if so, you will need to return its value (node\_data), and also remove that node by changing the value in frontOfList
* Otherwise, that means there are 2 or more nodes. Create a current “pointer” as we have done, setting it to start = to frontOfList. In a While loop, check if current’s pointer is pointing to a nodes whose pointer is null. If it is not, walk current down to point to the next node, and go back around thru the while loop. Your current pointer should not get to the last node! When it gets to the 2nd-to-last node, then you take action.
* When you exit the While loop, current should be equal to the node that is just BEFORE the last node. Which means you need to return the value (node\_data) of the **next node** and cut that last node free to be an orphan. (Cutting it loose takes 1 instruction.)