8.2

\* Code block #1:

Fix:

-Declaring the variable isn’t the same fisrt variable (nadePtr -> nodePtr).

-Declaring struct isn’t the same fisrt struct (ListNode -> listNode).

-There is no implementation of the next Node which is nullptr.

-The variable ‘nodePtr’ has not been declared ‘head’ to append Node.

Code after fix:

void NumberList::appendNode(double num)

{

    ListNode \*newNode, \*nodePtr;

    nodePtr = head;

    newNode = new ListNode;

    newNode->value = num;

    newNode->next = nullptr;

    if (!head)

        head = newNode;

    else

    {

        while (nodePtr->next)

            nodePtr = nodePtr->next;

        nodePtr->next = newNode;

    }

}

\* Code block #2:

Fix:

-When delete head, value in head will be lost. So we must add **ListNode pcru = head** to save value of head.

-We must declare the next node of nodePtr is nullptr and nodePtr is nullptr after delete.

Code after fix:

void NumberList::deleteNode(double num)

{

    ListNode \*nodePtr, \*previousNode;

    if (!head)

        return;

    if (head->value == num)

    {

        ListNode \*pcur = head;

        head = head->next;

        delete pcur;

    }

    else

    {

        nodePtr = head;

        while (nodePtr->value != num)

        {

            previousNode = nodePtr;

            nodePtr = nodePtr->next;

        }

        previousNode->next = nodePtr->next;

        nodePtr->next = nullptr;

        delete nodePtr;

        nodePtr = nullptr;

    }

}

\* Code block #3:

Fix:

-When delete all node after the first node, we must free the allocated space, because nullptr is the garbage for memory.

Code after fix:

NumberList::~Numberlist()

{

    ListNode \*nodePtr, \*nextNode;

    nodePtr = head;

    while (nodePtr != nullptr)

    {

        nextNode = nodePtr->next;

        nodePtr->next = nullptr;

        delete nodePtr;

        nodePtr = nextNode;

    }

}