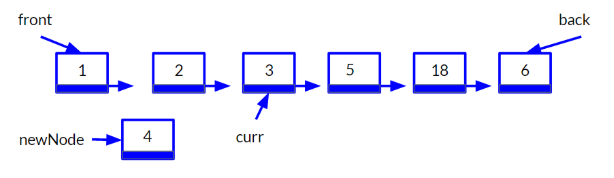
**Linked List Activity**

Use your LinkedList.h file from the No Brainer. Set the project up using the tester code below.

Consider the following Linked List.



Our goal is to add the value 4 into the list and remove the value 18 from the list, so our list shows as 1, 2, 3, 4, 5, 6

**Adding 4 to the List**

The pointer curr in the image above is currently pointing to the node with the value 3. To get the pointer to that position, it would start at the front of the list and iterate to that point. Follow the next few steps to get curr into that position and add 4 to the list

1. Look at the image above, what pointers would need to be modified in order to add 4 into the linked list in the correct order? **The next point for both the 3 and 4 node.**
2. This means we need to set these pointers up. Currently, the pointer curr is not in the code. Write the statement to declare curr and set the pointer curr to the front of the list.
   1. **Auto newNode = make\_shared<Node<Type>>(4);**
3. Write a while loop to move the pointer to the correct position as shown in the image.
   1. **Auto curr = front;**
   2. **While (curr->next->data < info) {**
      1. **Curr = curr->next;**
   3. **}**
4. Write the cout statement that would use the curr pointer to display the 3 to make sure it is working.
   1. **Cout << curr->data;**
5. Look at the image. Discuss what pointers would need to be modified to add the newNode into the correct position.
   1. **The pointer of the “3” node would need to be modified to be pointing to the newNode, and the newNode’s next pointer would need to be set to the “5” node. You would have to do the newNode’s pointer first though, otherwise the “5” node would be lost.**
6. Modify those pointers. Run the program to be sure it displays 1, 2, 3, 4, 5, 18, 6
   1. **newNode->next = curr->next;**
   2. **curr->next = newNode;**

**Removing 18 from the list**

Next, we are going to delete 18 from the list. Remember, the purpose of this is to practice using pointers, therefore we will accomplish this by modifying the pointers and we will **not** call the remove() method. Follow the steps below.

1. We will need to move the pointer named curr to the Node right before 18. Write a loop to push curr to the Node with the value 5. Add a cout statement for the variable curr to be sure it now displays the value 5.
   1. **auto curr = front;**
   2. **While (currVal->next->data < info) {**
      1. **curr = curr->next;**
   3. **}**
   4. **cout << curr->data;**
2. It will be helpful to have a temp pointer to point to the value we want to delete. Write to set temp equal to the Node with the value 18. (You can use the curr pointer to accomplish this)
   1. **auto temp = curr->next;**
3. Write the statement that would remove the Node with the value 18 by setting the next value of the pointer curr to the next value of temp.
   1. **curr->next = temp->next;**
4. Run the program and be sure it is displaying 1, 2, 3, 4, 5, 6

**Tester Code**

We will create a method to use for this activity called pointer practice. Add the following to your LinkedList class to test out your code

Add a function prototype to the class

void pointerPractice(Type, Type);

Then add the following method in the .h file after the class:

template<class Type>

void LinkedList<Type>::pointerPractice(Type toInsert, Type toRemove)

{

auto newNode = make\_shared< Node<Type>>();

newNode->data = toInsert;

}

**Main**

Use the following main() file to match the data

#include <iostream>

#include "LinkedList.h"

using namespace std;

int main()

{

LinkedList<int>numbers;

numbers.insert(1);

numbers.insert(2);

numbers.insert(3);

numbers.insert(5);

numbers.insert(18);

numbers.insert(6);

numbers.pointerPractice(4, 18);

cout << "List Values: " << numbers << endl;

return 0;

}