Individual Role Breakdown

Jack Bender is Leader

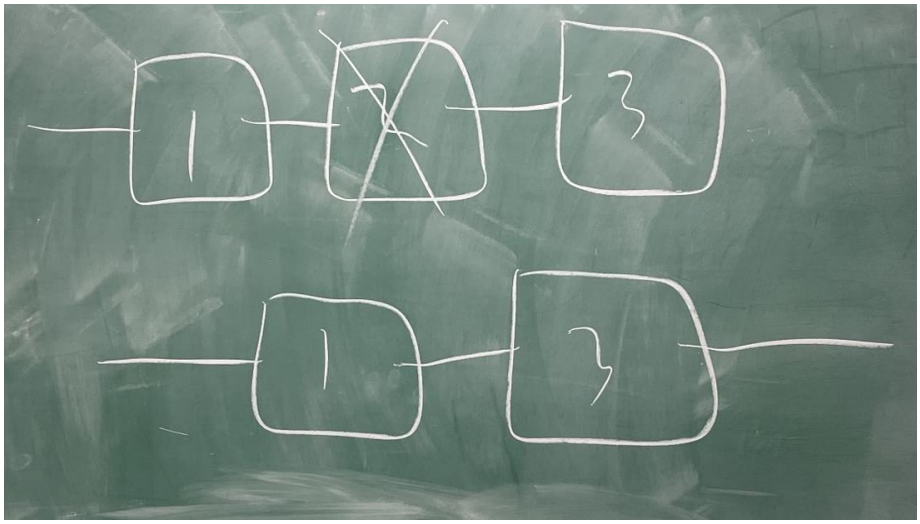
Cole Wilson is Architect

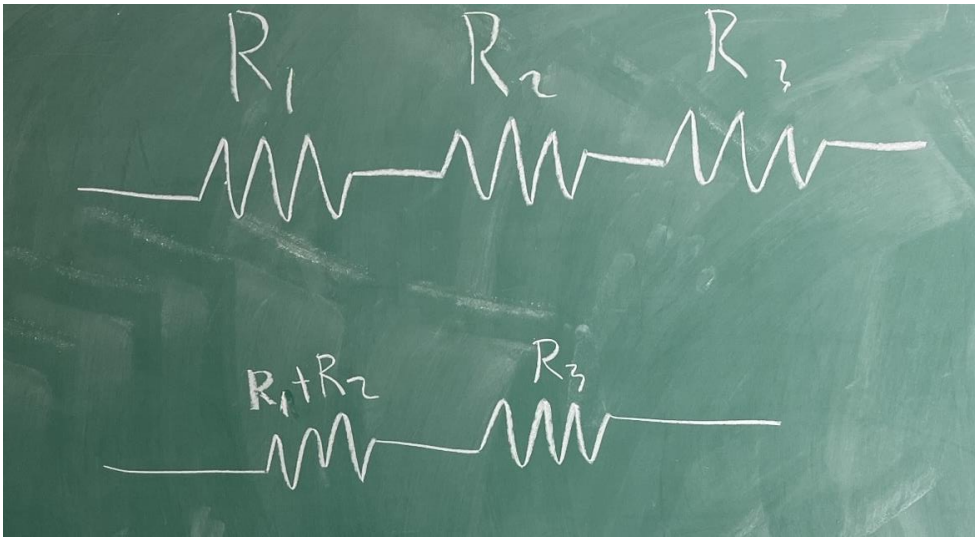
Program Idea

We designed our linked list so that it can analyze resistor series circuits (we are both Electrical Engineers). When reducing resistors in series you just add them together, so we can achieve some circuit simplification functionality using the pop function.

I attached a couple pictures for people who do not know circuits:

We can use the pop function to achieve this functionality (though it would be crossing out 1 instead of 2):





How TAs will Run/Test the Program

We have included a makefile. Just type “make” into the terminal and an executable named “output” will be made.

Alternatively, you can compile using g++ *.cpp and run the executable (a.exe).

Then using the test case file:

a.exe <TEST_CASE.txt

Classes Used

LinkedList, ListNode, Data

Functions Implemented

Driver Function:

1. Main: runs the user interface and calls other functions as needed

Linked List Functions:

1. Is Empty: Check if the List is empty
2. Get Length: get the length of list (how many resistors)
3. Append: add resistor to end of list
4. Prepend: add resistor to start of list
5. Insert: add resistor to specific spot in list
6. Get Front: gets the value of the resistor at the front of the list
7. Remove: remove all of one kind of resistor from list
8. Pop: remove the resistor from the head of the list
9. Display List: shows user all of the items in the list
10. Sort: sorts the list forward or backwards
11. Simplify Circuit: stores values and pops, then makes one new node

List Node Functions:

1. get Data: return data stored
2. set Data: simple function that puts a value into list at that point
3. set Next: simple function that puts a value into list at the next point
4. set Previous: simple function that puts a value into list at the previous point

Data Functions:

1. userInputInfo: get info from user to store in class and later list
2. deleteMemory: manages unwanted data when resistor removed from the list
3. plus the getters/setters

Program Flow

High level view of what the program does.

In driver there is a switch statement that allows the user to navigate a menu to pop, add, combine resistors, etc. The driver is where most of the user interface is. The program utilizes a doubly linked list for flexibility in functionality. The Linked List class acts as a circuit of resistors in series holding them in order, the Data class stores info about resistors like resistance and power rating, the ListNode Class provides a way to iterate through the list and view individual resistors.

Process

Cole made the GitHub and did a lot of the initial commits to outline what we needed. He primarily worked on the project write-ups, README, Data Class, Data functions, and the Driver. The Data class was not too hard since it was like the previous program, and he got the syntax for the comparison and stream operator overloads from the lecture notes and some of the lecture assignments. The Driver was likewise similar to the last program in that it used a do-while loop and some switch statements.

Jack primarily worked on the LinkedList and ListNode class, often referencing example code from lecture to get an idea of what to do. These classes were a lot more intricate due to how many functions were needed, and it was confusing trying to figure out how to make them interact with each other and Data class using templates. He asked the TAs for more information, and they were able to help him a bit. Once he knew what he needed to do with the templates, it was not too difficult to code the rest of these classes.

Results/Outcome

The results of this program could have been better, but they were not bad. We feel really good about the simplification of the circuit. The allocation of memory was difficult, and we may have data leaks but with our time frame, we could fix that. Overall, our program runs and works accordingly. We are propud of our project and have many ideas of how we could update it.

Thoughts/Comments/Concerns

This program was pretty fun to brainstorm since we were able to make it into something we both enjoyed and is applicable to our major. The sorting function itself was difficult, and the template helper class was pretty weird.