# Lab Week 7

## Pointers and Linked Lists

Skills Required

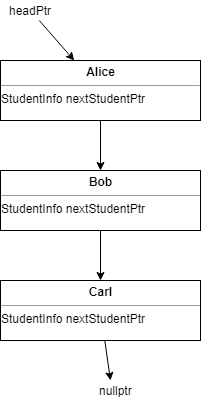
* Make a pointer reference, add a new reference, delete a reference, create and work with linked lists.
* Create and use classes, Exception Handling, Read and write Files, Work with arrays and vectors, Create Functions, Include Headers and other files, Loops (while, for), Conditional (if, switch), Datatypes, assignment, etc.
* Basic git commands

1. Follow the link that was given on Canvas for this Assignment to create your repository.
2. Use the directions and skills from Week 1 to clone the repository into the directory of your choosing.
3. Open the provided solution in the cloned folder.

**Assignment**

It’s very easy to make a small mistake in a linked list. You may want to review a first linked list in your book before beginning.

A linked list would look something like the following structure.



The headPtr variable points to the first object of StudentInfo, from there we can use the nextStudentPtr to go to the next student. We can simply set a variable to headPointer and keep going to the next item until our variable is nullptr. (Be very careful when you update headPtr, it is very easy to lose the reference to an item.)

StudentInfo\* headPtr = new StudentInfo("Alice");

StudentInfo\* temp = new StudentInfo("Bob");

headPtr->setNextStudent(temp);

Your program will read in a text file that has 2 values on each line the first is the name, the second is the action. Possible actions are add for add to our linked list and del for delete form our linked list. The most difficult example file you are given is input7.txt.

carl add

alice add

bob add

donna add

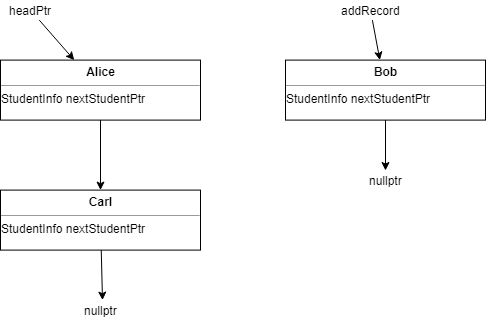
bob del

alan add

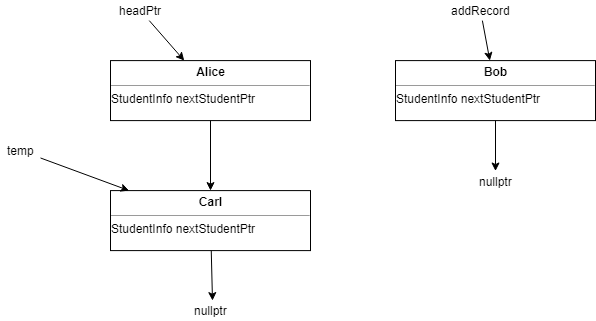
alice del

marco add

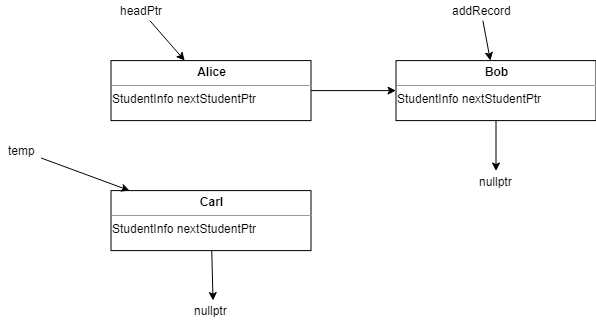
As you read each line from the text file you will build up your linked list. We don’t just want to add each item to the end of our linked list (although you may want to start out that way). Instead we are going to insert the student in position in the list that they belong **alphabetically**. Alice is before Bob and Bob is before Carol. (We won’t worry about case). If our linked list is alice->carl and we read bob from the file, then we have to insert bob in between alice and carl. In order to preserve that we need to create temporary pointer to the student alice points to next (carl) and then make alices next ptr point to bob, then set bob’s next pointer to our temp pointer which is carl.



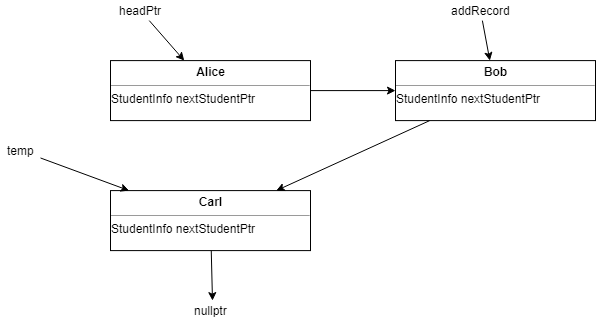
Step 1. Set a temp variable to where alices next pointer is.



Step 2. Redirect alice pointer to our new added record



Step 3. Set bobs next pointer to the temp pointer



If the new record is less than our headPtr record then we need to set the new records pointer to the headPtr and update the headPtr to be our new record. I’ll let you work that out in for yourself.

Another harder step is going to be deleting a record. If we have Alice->Bob->Carl->Donna and we want to delete carl, we can start by going through our linked list starting from the head pointer. If our current record is Carl, we can’t just delete Carl though. If we do, then Bob will have a next pointer that is invalid and we’ve lost any reference to Donna. Work your way through the linked list, if the next item after the current one is the one to delete, then you can make a temporary pointer to that next record ( the one we want to delete), set the current next reference to the next pointer of the record we are deleting. We’ve then snipped out the record we want to delete, but we haven’t released the memory yet. Make sure you use delete to get rid of the reference.

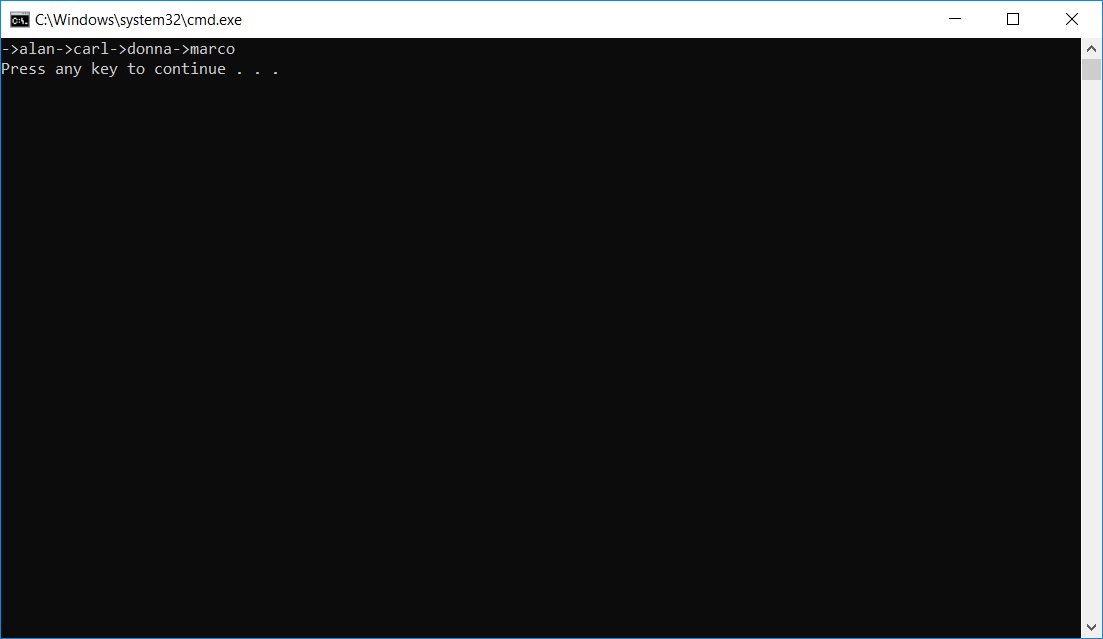
With those scenarios out of the way, you will read through a given text file and either add or remove the name from the linked list in alphabetical order. You’re given many files to help you test, starting with an easy scenario. Each one tests a different problem that you’ll want to consider. The last input file is the toughest with all the scenarios combined. Just change the file you are reading to a new input and test each one individually as you work.

|  |  |
| --- | --- |
| File | Scenario |
| input1.txt | All items are add, and they are in order. So you can just add each one to the end of the linked list. |
| input2.txt | An item will have to be inserted to the middle of the list somewhere. |
| input3.txt | Item(s) will have to be inserted at the head of the list |
| input4.txt | Item will be removed from a list |
| input5.txt | Item is removed at the end of a list |
| input6.txt | Beginning item is removed |
| input7.txt | Combination of all the scenarios |

Each scenario is different and is given because it may represent a problem that is easy to overlook.

Once you have read in the file and created the linked list you should output it as shown in the example output for input7. After that make sure you release any memory. (You can’t just delete the record at the head pointer because that simply orphans the rest of the linked list.)

Output for input7



**Above and beyond.**

It would be nice to compare the 2 instances instead of comparing the names of the 2 instances. You may want create an overload for the == and < comparisons. Try to add those to your program.

<https://www.learncpp.com/cpp-tutorial/96-overloading-the-comparison-operators/>

**Submit your assignment**

1. Update your files on GitHub (Remember to save the files in the IDE before adding them) .sln, .cpp, .h, .vcxproj and any others.
   1. use git status to check which files have been changed and need to be staged
   2. use git add to add the files that have been added or changed
   3. use git commit -m “message” to commit to your local repository
   4. use git push to push this to the remote repository so the grader can find and validate it.
2. Check github repository that your changes and submission are correct. **Show the lab instructor to have them validate the submission.**