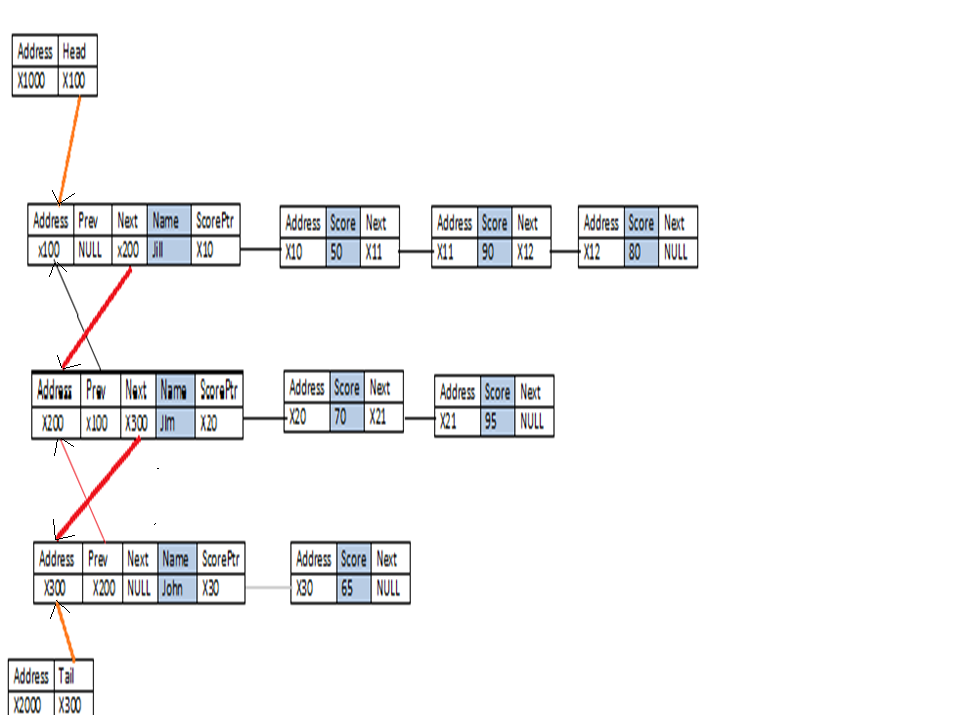
**Two linked lists**

Write a program that reads a list of students (first names only) from a file. It is possible for the names to be in unsorted order in the file but they have to be placed in sorted order within the linked list.

The program should use a **doubly linked list**.

Each node in the doubly linked list should have the student’s name, a pointer to the next student, a pointer to the previous student and a pointer to a single linked list that will contain each student’s set of scores. Here is a sample visual. The head points to the beginning of the list. The tail points to the end of the list.



struct scoreNode{

char score[10];

scoreNode \*next;

};

typedef struct scoreNode scoreNode;

struct studentNode{

studentNode \*prev;

studentNode \*next;

char name[30];

scoreNode \*scoreFront;

scoreNode \*scoreRear;

};

int main()

{

char data[30];

studentNode \*head = NULL, \*tail = NULL, \*curr;

bool done;

while(clean() &&input.good()){

input >> data;

if(strcmp(data,"delete")!=0) {

curr=insertStudent(data,head,tail);

done=false;

do{

input >> data;

if (strcmp(data,"end")==0) //end means there are no more scores

done=true;

else

enqueueScore(data, curr->scoreFront, curr->scoreRear); }while (!done);

}else{ //delete a student

input >> data; //name of student to delete

deleteStudent(data,head,tail); //delete student

}

}

outputStudent(head,tail);

}

When inserting consider all the following conditions:

if(!head){ //no other nodes

}else if (strcmp(data, head->name)<0){ //smaller than head

}else if (strcmp(data, tail->name)>0){ //larger than tail

}else{ //somewhere in the middle

}

When deleting a student consider all the following conditions:

student may be at the head, the tail or in the middle

The information for each student will be on a single line in a text file. Below, you will find a sample of what the file looks like. Notice the names are in unsorted order but the information placed in the linked list (above visual) is in sorted order. The name of the file should be “input.txt”. Also, notice the scores for each student ends with the word “end”.

In the text file, the word delete followed by a name, should delete the node with that specific student’s name from the doubly linked list. If the name is not found, then nothing is deleted.

(NOTE: The above visual represents only the first three lines from the text file below.)

Jim 70 95 end

jill 50 90 80 end

John 65 end

**delete jill**

**jack 50 35 end**

**Jane end**

**delete jim**

At the end of the program, traverse through the contents of the linked lists in both ascending and descending order, using the doubly linked list, and write the contents into the file output.txt. For example, given the above list, here is the sample display:

John 65

Jack 50 35

=============

Jack 50 35

John 65

Place the following right after your #includes

ifstream input("input.txt");

ofstream out("output.txt");

Assumptions: Names are all in lower case. Names cannot be duplicated.

HINTS:

When inserting, there are a number of conditions to keep in mind:

1) You can insert a name into an empty list (head and tail are both null)

2) If the list contains something, a name can be inserted before the head, which means you have to update the head.

3) If the list contains something, a name can be inserted after the tail, which means you need to update the tail.

4) If the list contains something, a name can be inserted somewhere in the middle, which means you need to traverse the list, using some kind of loop.

Also, deletes can happen from the beginning, the end or somewhere in the middle. Make sure to check all the various conditions.

For the single linked list of scores, you can use a queue, which is a FIFO structure.  It is possible for a student to have no scores.

Summary:

When deleting, make sure to deallocate all the spaces. This would be all the scores for the individual from the singly linked list and the individual from the doubly linked list.

There is no user input. Everything is done through the file.

* Code should be modularized.
* Comment your code.
* Make sure that your code is properly indented.
* Do not use classes. Write the program in C.
* The only global variables should be the file handles.
* The name of your input file should be **input.txt.**
* The name of your output file should be **output.txt**. Do not output to the screen. Write everything into the file.
* The names in the text file may appear in unsorted order but should be inserted into the linked list in sorted order.
* “end” marks the end of the list of scores for each student.
* “delete”, followed by a name, should eliminate that individual, along with his/her scores from the list.
* The list should be written in ascending and descending order, with a line in-between, into the file, output.txt.

**Caution**: All parties involved in the act of plagiarism will receive -50 points on the assignment. No distinction will be made between those who copy and those who allow their work to be copied. All points awarded for the group assignments that pertain to this homework assignment will be taken away. Also, you will forfeit getting an A in the class even if you have enough points. If you are caught plagiarizing, you will be reported to the Dean of student services.

NO EXCEPTIONS!!!!!!