CSC 250 - Program 2 Part 1

Word Frequency – Linked Lists

Assigned 3/2/2015 Due: 3/25/2015

**Description:**

A book editor is wanting to view the Frequency that words are used in a short story. He suspects that truly good short stories use very few words over and over. He has no proof of this and would like your help. You will read in all the words from a short story and count the number of times that a particular word occurs. After processing the entire file, you will then output the all words that occurred n number of times followed by the all the words that occurred the next number of times and so on.

**Command Line:**

Your program will be started on the command line. You must error check all options that are provided. If the user fails to enter the proper number of command line arguments, output an appropriate error message, print the usage statement and exit the program.

C:\> prog2.exe shortstory.txt results.txt

The first argument will be a text file that contains the short story. The second argument is the name of the file where you will output your results to. If either file fails to open, output and error message informing the user which file failed to open and exit the program.

**Input File:**

The input file is just a text file. You will read in one word at a time. Each word will be of the stl string type. You will need to remove all punctuation from the front and end of the word. Do not worry about punctuation that occurs in the middle. Next, convert the string to lower case. After preparing the word, you will insert it into the list of words if it does not already exist. If the word is already in the list, you will increment the frequency counter of the word. You will continue to do this for every word in the file.

Examples: “Don’t!” Becomes don’t

“…Shout” become shout

**Output File:**

After you have processed the entire short story, you will output the words that occurred the most in the story first followed by the next most popular words. You will put a nice header to the file that displays the frequency count. This header must be spotted easily within the file. Below the header will be the words that occurred a particular amount of time. They will be in order according to the ascii values.

Example:

**===============================================================================**

**Frequency Count: 13**

**===============================================================================**

**as me**

**merchant that**

**===============================================================================**

**Frequency Count: 12**

**===============================================================================**

**on**

**===============================================================================**

**Frequency Count: 10**

**===============================================================================**

**but castle**

**my**

Notes:

You can clearly pick out the frequency header in this file. The words are nicely formatted and you may assume no word will be over 30 characters in length for formatting purposes.

As, me, merchant and that all occurred 13 times in the short story.

On occurred 12 times in the story.

But, castle, and my occurred 10 times in the story.

**The Project:**

Your team will implement a multi file project. Each team member is expected to do their share of the programming. A team member may be asked “what is going on in this section of code?” to validate that they are doing their fair share of the work in the event of a team is not working out. When this happens, you may be asked to code the program by yourself and you will lose 20 points for not working with your teammates. The functions in your program will be documented and authors is important. I would also like you to document who helped Debug/test each function.

Prog2.cpp – will contain main.cpp and any functions not related to the linklist class. Prototypes for these functions will go at the top of prog2.cpp just like in csc150.

Linklist.h – will contain the class definition of the linklist class described below.

Linklist.cpp – will contain the implementation of the **linklist class only**. I would recommend that each team member have a file named after their id and when the program is working, copy paste the contents over into the linklist.cpp file. This makes it very easy to update functions that you didn’t write.

1035687.cpp

7034519.cpp

1059384.cpp

1093434.cpp

**Implementing the list:**

You may not change the order of the elements below. If you wish to add a function, you must get prior approval from you instructor. Be prepared to defend why you need the function. You may not change what each function does. Your algorithm may not use all of the functions, but they all must be coded and word correctly.

class LinkList

{

    public:

        LinkList();

        ~LinkList();

        bool insert( string word );

        bool remove( string word );

        bool find( string word );

        bool incrementFrequency( string word );

        bool isEmpty();

        int getMaxFrequency();

        int size();

        void print( ostream &out );

    private:

        struct node

        {

            int frequencyCount;

            string word;

            node \*next;

        };

        node \*headptr;

};

The insert Function – will enter the given word into the list in ascending order based on the word value. After insertion, it will set the frequency count to 1. If a duplicate word is passed in, you will insert the duplicate word into the list and set its frequency to 1. **This function does not increment the frequencies of the words at all.** The function will return true for inserting a word or false it could not insert the word.

The remove Function – will erase the given word from the list if it is located. If the word is removed, the function will return true. If the word is not found, the function will return false. The function will remove only the first occurrence of the word.

The find function – will traverse the list looking for the requested word. If the word is located, the function will return true. If the word is not located within the list, the function will return false.

The Increment Frequency function – will traverse the list looking for a particular word. If the word is located, the frequency count for that word will be incremented by one.

The Is Empty function – will return true if nothing is in the list or false if the list contains items.

The getMaxFrequency() – will return the frequency Count of the most popular word. If the list is empty, a zero is returned.

The size function – will tell you how many words are in your list.

The print Function – will output the words from highest frequency to lowest frequency as described above to the ostream given.

OTHERS: other functionality may be added to the class if you seek prior approval and can justify why it is needed.

**Algorithm**:

1. Validate the existence of command line arguments
2. Validate opening of the 2 files
3. While you have not read in all words from the file, get a word
   1. If the word is in the list already, increment the frequency
   2. If the word is not in the list, add the word to the list
   3. Go to step 3 until file is processed
4. Output the words to the file

CSC 250 - Program 2 Part 2

Word Frequency – STL Lists

Assigned 3/2/2015 Due: 3/25/2015

**Task:**

You will rewrite same program using the stl list class. To use this class, you must include the <list> library. This library can hold any item needed. We will have it hold a structure.

struct item

{

    int frequencyCount;

    string word;

};

**To declare a list of items:**

List<item> theList;

Functions you will find useful:

Find() – located in algorithms

Insert() – member function of the list class. To insert an item, you will need to pass in a structure that is already populated with data.

item theItem;

theItem.word = aword;

theItem.frequencyCount = 1;

Begin() – member function of the list class

End() – member function of the list class

Sort() – member function of the list class. Must provide a function that compares 2 items. This function returns true if the first argument goes before the second argument and false otherwise.

bool compare2Items( node &l, node &r)

{

// your code for ordering here

}

Note: Sorting allows you to reorder the list based on the frequencies so you can output a header when the count changes.

This is a one file program and should be called prog2stl.cpp

**TimeLine:**

Your List – Part 1

3/2 Get your partners name and contact info.

3/4 Validate command line arguments, opening files, and reading words till end of file

3/6 Preparing each word for inserting into link list

3/7-3/15 Enjoy Spring Break (read up on the stl List class)

3/16 Constructors, destructors, isEmpty, and a simple print function written

3/17 Have the insert and find functions working

3/18 have the incrementFrequency, size, getMaxFrequency working

3/19 Rewrite the print function to output the list as requested above and finish work on prog2.cpp

STL – Part 2

3/20 Rewrite the program using the stl List class. Make a copy of your prog2.cpp file and remove references to your linklist class. The argc, argv, files will all be valid.

3/21 add the words to the file in the appropriate location if the word is not in the list

3/23 Increment the word counter if the word is in the list. Processing all words in the file.

3/24 Be able to output list to file using same rules as first part.

3/25 (Optional) Sort the list based on frequency counter, if words have same count, then use the word value.

3/25 Program is due at midnight. (11:59 PM)

**Helpful Hits:**

Don't panic, work on small sections at a time and don't be afraid to write functions.

Save and compile often.

Work on the project according to the time line.

Don't wait until the last minute:

If you have problems, try to figure it out, but don't be afraid to get help

Seek help from your teammates first. If you are coming to me, you must have one of your team mates with you.

Work on your program frequently, trying to do large pieces of your program at one time often leads to careless mistakes.

If your team is having problems with a team member, keep me informed. If I do not know a problem exists, I cannot help resolve it.

The majority of points will be in the first part of the program. Make sure it is working before moving onto part 2.

I do not debug programs on the day they are due.

Submit your program on the due date by zipping the following 4 files (prog2stl.cpp, prog2.cpp, linklist.cpp and linklist.h). After submitting your programs, each member will fill out the team evaluation sheet. This will not affect the grades of your teammates but is used for ABET certification. Please be honest.

**You must work on a team. Failure to meet with the team can result in 50 points being subtracted from your grade. Keep me informed of any problems you have with team members.**