**Homework #2: string input, struct, vectors, parameter passing and file I/Os**

**The main goal** of this homework is to count and sort all the unique tokens in a file by applying the following knowledge points: reading and writing a text file, creating a new data type using *struct*, managing a vector of *struct* objects, passing a vector into and out of a function, understanding and implementing the insertion sort and selection sort algorithms.  **Problem description:**  
Steve Jobs delivered a touching and inspiring speech at Stanford's 2005 commencement. The transcript of this speech is attached at the end of this homework description. In this homework, you are going to write a program to find out all the unique tokens (or words) used in this speech and their corresponding frequencies, where the frequency of a word *w* is the total number of times that *w* appears in the speech.  You are required to store such frequency information into a vector and then sort these tokens according to frequency.  Please feel free to use existing functions such as strtok() or sstream to identify tokens in this implementation.

Specifically,  you are required to include the following elements in your program:

* Declare a *struct TokenFreq* that consists of two data members: (1) *string value*;  and (2) *int freq;*  Obviously, an object of this struct will be used to store a specific token and its frequency.  For example,  the following object *word* stores the token "dream" and its frequency 100:

*TokenFreq word;*

*word.value="dream";*

*word.freq=100;*

* + Remember to declare this struct at the beginning of your program and outside any function. A good place would be right after the "using namespace std;" line. This way, all the functions in your program will be able to use this struct to declare variables.  (**5 points**)
* Implement the function *vector<TokenFreq> getTokenFreq( string  inFile\_name);* This function reads the specified input file line by line, identifies all the unique tokens in the file and the frequency of each token. It stores all the identified (token, freq) pairs in a vector and returns this vector to the calling function. Don't forget to close the file before exiting the function. **In this homework, these tokens are case insensitive.** For example, "Hello" and "hello" are considered to be the same token.  **(20 points)**
* Implement the selection sort algorithm to sort a vector<TokenFreq> in ascending order of token frequency. The pseudo code of the selection algorithm can be found at <http://www.algolist.net/Algorithms/Sorting/Selection_sort> You can also watch an animation of the sorting process at <http://visualgo.net/sorting> -->under "select".  This function has the following prototype:
  + - *void selectionSort( vector<TokenFreq> & tokFreqVector );* This function receives a vector of TokenFreq objects by reference and applies the selections sort algorithm to sort this vector in increasing order of token frequencies.  **(20 points)**
* Implement the insertion sort algorithm to sort a vector<TokenFreq> in descending order of token frequency. The pseudo code of the selection algorithm can be found at <http://www.algolist.net/Algorithms/Sorting/Insertion_sort> Use the same link above to watch an animation of this algorithm.  This function has the following prototype:
  + - *void insertionSort( vector<TokenFreq> & tokFreqVector );*  **(20 points)**
* Implement the *void writeToFile( vector<TokenFreq> &tokFreqV, string outFileName);* function.  This function receives a vector of TokenFreq objects and writes each token and its frequency on a separate line in the specified output file.  **(10 points)**
* Implement the *int main()* function to contain the following features: (1) asks the enduser of your program to specify the name of the input file, (2) ) call the *getTokenFreq()* to identify each unique token and its frequency, (3) call your selection sort  and insertion sort  functions to sort the vector of TokenFreq objects assembled in (2); and (4) call the *WriteToFile()* function to print out the sorted vectors in two separate files, one in ascending order and the other in descending order.      **(15 points)**

**Example input and outputs:**

Assume that your input file contains the following paragraph: ﻿"And no, I'm not a walking C++ dictionary. I do not keep every technical detail in my head at all times. If I did that, I would be a much poorer programmer. I do keep the main points straight in my head most of the time, and I do know where to find the details when I need them. by Bjarne Stroustrup"

After having called the getTokenFreq() function, you should identify the following list of (token, freq) pairs and store them in a vector (note that the order might be different from yours):  {'no,': 1, 'and': 1, 'walking': 1, 'be': 1, 'dictionary.': 1, 'Bjarne': 1, 'all': 1, 'need': 1, 'Stroustrup': 1, 'at': 1, 'times.': 1, 'in': 2, 'programmer.': 1, 'where': 1, 'find': 1, 'that,': 1, 'would': 1, 'when': 1, 'detail': 1, 'time,': 1, 'to': 1, 'much': 1, 'details': 1, 'main': 1, 'do': 3, 'head': 2, 'I': 6, 'C++': 1, 'poorer': 1, 'most': 1, 'every': 1, 'a': 2, 'not': 2, "I'm": 1, 'by': 1, 'And': 1, 'did': 1, 'of': 1, 'straight': 1, 'know': 1, 'keep': 2, 'technical': 1, 'points': 1, 'them.': 1, 'the': 3, 'my': 2, 'If': 1}

After having called the selectionSort() function, the sorted vector of token-freq pairs will contain the following information (again, the tokens of the same frequency might appear in different order from yours) : [('no,', 1), ('and', 1), ('walking', 1), ('be', 1), ('dictionary.', 1), ('Bjarne', 1), ('all', 1), ('need', 1), ('Stroustrup', 1), ('at', 1), ('times.', 1), ('programmer.', 1), ('where', 1), ('find', 1), ('that,', 1), ('would', 1), ('when', 1), ('detail', 1), ('time,', 1), ('to', 1), ('much', 1), ('details', 1), ('main', 1), ('C++', 1), ('poorer', 1), ('most', 1), ('every', 1), ("I'm", 1), ('by', 1), ('And', 1), ('did', 1), ('of', 1), ('straight', 1), ('know', 1), ('technical', 1), ('points', 1), ('them.', 1), ('If', 1), ('in', 2), ('head', 2), ('a', 2), ('not', 2), ('keep', 2), ('my', 2), ('do', 3), ('the', 3), ('I', 6)]

**Grading:** This homework will be graded for a total of 90 points.

**Submission instructions:** Submit **a single .cpp file** on iLearn.