Homework Assignment

Class:	CS202	**Semester:**	Fall 2018
Assignment type:	Homework assignment	**Due date:**	9/7/18
Assignment topic:	CS135 - Data manipulation	**Assignment no.**	1
Delivery:	WebCampus – cpp file		

Goal

Practice the file I/O, arrays, data manipulation and parameter passing.

Input to the program

- Input data is provided in the file *elections.txt*. Download from WebCampus.
- Input format for each line: F=*firstname*,L=*lastname*,*V=votes*.
- Read only valid lines into the array

Procedure for the implementation

- Read sections *Goal*, *Input to the program*, *Output* to get the overall picture of the assignment. Implement the assignment step by step, testing each step before proceeding to the next one.
- Declare array of structs (size ***ARR_SIZE***):

```
struct Candidate {
  string first;
  string last;
  int votes;
  double pScore;
};
```

- Read candidates' names from the file elections.txt - implement *readFile(Candidate[])*
- Implement *displayList*(*Candidate[]*) function
- Implement *displayCandidate*(*Candidate*) function
- Implement *getWinner*(*Candidate[]*) function
- Implement *getLast*(*Candidate[]*) function
- Implement *calculateScores*(*Candidate[]*)
- Implement *sortByVotes*(*Candidate[]*) function
- Implement *roundScore*(*Candidate &*)

Output

- Display candidates' names using *displayList*() function
- Execute the *getWinner*(*Candidate*[]) function to get the candidate with the highest number of votes. Display his name along with number of votes.
- Execute the *getLast*(*Candidate*[]) function to get the candidate with the lowest number of votes. Display his name along with number of votes.
- Calculate *pScore* for each candidate
- Sort candidates by votes
- Display sorted list using *displayList*() function
- For three records with the highest *pScore*, use the *roundScore* function to round the *pScore*
- Display list again using *displayList*() function

Functions specification & definitions:

- ***ARR_SIZE* –** size of the array. It is assumed that number of candidates in file will never exceed the ***ARR_SIZE***. Hardcode ***ARR_SIZE*** to 100.
- When iterating over the candidates' list, do not iterate over the entire array, but just over the records where data is filled in.
- ***score (votes)*** – number of votes earned by candidate
- ***pScore*** – percentage score calculated using formula (1)
- ***void readFile*(*Candidate candidates*[])** – reads the *elections.txt* file, fills the *candidates[]* array. Hint: use *substr*() and *find*() functions. Set *pScore* to 0.
- ***void displayList*(*Candidate candidates*[])** – prints the array of *Candidate* structs. One candidate per one line, include all fields. Use *setw()* to display nice looking list.
- ***void displayCandidate(Candidate cand)* –** prints the complete information about the candidate.
- ***Candidate getWinner*(*Candidate candidates*[])** – returns single struct element: candidate with highest score
- ***Candidate getLast*(*Candidate candidates*[])** – returns single struct element: candidate with lowest score
- ***void sortByVotes*(*Candidate candidates*[])** – function sorts the *candidates[]* array by number of votes, the order in candidates[] array is replaced
- ***void calculateScores*(*Candidate candidates*[])** – calculates the percentage score for each candidate. Use the following formula:

$$pScore = \frac{s_c}{\sum_{c=1}^{C} s_c} \cdot 100\% \qquad (1)$$

- ***void roundScore(Candidate &cand)*** – updates single element, passed by reference. Function is rounding the *pScore* (example: 74.29% is rounded to 74%, 74.64% is rounded to 75%).

s_c – score for candidate c
C – number of candidates

Code skeleton

```
#include <iostream>
#include <fstream>
#include <string>
#include <stdlib.h>
#include <iomanip>

using namespace std;

const int ARR_SIZE = 100;

void readFile(Candidate[]);
void displayList(Candidate[]);
void sortByVotes(Candidate[]);
void displayCandidate(Candidate);
Candidate getWinner(Candidate[]);
Candidate getLast(Candidate[]);
void calculateScores(Candidate[]);

int main() {

}

void readFile(Candidate candidates[]) {
    string line;
    ifstream infile;
    infile.open("elections.txt");
    while (!infile.eof()) {
        getline(infile,line);
        // your code here
    }
    infile.close();
}

void displayList(Candidate candidates[]) {
}

void sortByVotes(Candidate candidates[]) {
}

void displayCandidate(Candidate cand) {
}

Candidate getWinner(Candidate candidates[]) {
}

Candidate getLast(Candidate candidates[]) {
}

void calculateScores(Candidate candidates[]) {
}
```

Sample output

```
ALL CANDIDATES:
  First:      Last:     Votes:     % Score:
   John      Smith       3342           0%
   Mary       Blue       2003           0%
   Bill     Warren       1988           0%
 Robert     Powell       5332           0%
   Stan      Smith       4429           0%
  Laura       Rose       3392           0%

ALL CANDIDATES:
  First:      Last:     Votes:     % Score:
   Bill     Warren       1988          9.70%
   Mary       Blue       2003          9.77%
   John      Smith       3342         16.31%
  Laura       Rose       3392         16.55%
   Stan      Smith       4429         21.61%
 Robert     Powell       5332         26.02%

winner:
FIRST NAME: Robert
LAST NAME:  Powell
VOTES:      5332
% GAINED:   26%

lowest score:
FIRST NAME: Bill
LAST NAME:  Warren
VOTES:      1988
% GAINED:   9%


ALL CANDIDATES:
  First:      Last:     Votes:     % Score:
   Bill     Warren       1988          9.70%
   Mary       Blue       2003          9.77%
   John      Smith       3342         16.31%
  Laura       Rose       3392         17%
   Stan      Smith       4429         22%
 Robert     Powell       5332         26%
```

Submission:

Include the following elements in your submission: (rid = your rebel id)

Problem	Element	File
1	Code of your program	rid_1.cpp file
	Summary of the submission	
	Summary: 1 file, submit it to the WebCampus. Remember about proper names of the files!	