Project 2 (Part A): STL Candidate List

Due date:

- **MW class** ➔Monday, May 2, at the beginning of class
- **TTh class** ➔ Tuesday, May 3, at the beginning of class

Project name: A250_P2A_Teamname

After making all the necessary corrections on **Project 1**, change the implementation so that the list of candidates is an **STL vector** instead of a linked list.

NOTE: If you had **errors** in **Project 1**, you must fix them. Each error **will count 1 pt.** if it is still present in **Project 2**.

SECTION 1 – Adding an isEmpty function

Instead of having multiple errors messages in the **CandidateList** class, you will be checking if the list is empty from the **processChoice** function in the **Main.cpp** file, **before** calling any functions.

Implement the function **isEmpty** in the **CandidateList** class so that it returns true if the vector is empty and false otherwise.

Modify all the selections in the **processChoice** function of the **Main.cpp as follows:**

- Each choice should check if the list is empty
    - If the list is empty, return an error message (the error message should be the one that is implemented in the member function(s) you are calling
    - Otherwise, call the necessary function(s)

Remove all error messages indicating that the list is empty from all member functions of the **CandidateList** class (when you do this, make sure you **indent the code appropriately** and you do **not leave unnecessary white space**).

To test all selections and make sure you are getting the correct display, comment out the call **readCandidateData(candidateList);** in the **main( )** function, and compare your output against the **output_empty_list** executable file. Test each selection.

SECTION 2 – The STL <vector>

In this section, you will change the way the list of candidates is represented by storing all candidates in an **STL vector** instead of a linked list.

Modify the following:

- **CandidateList.h**
  - Include the **STL <vector>** class.
  - Delete the class **Node**.
  - **Member variables:** You will not need the first and last pointer any longer, and no need for the variable count either. Delete all member variables and declare an **STL vector** of **CandidateType** named **candidates**.
  - Function **destroyList**: You have no dynamic variables; therefore, you can delete this function.
- **NOTE:** Before you move onto modifying the implementation of the class, make sure you keep in mind the following:
  - Function **size** of the **STL <vector>** class does **NOT** return an **int**; therefore, you will need to cast the function using **static_cast**.
  - You will be eliminating several **parentheses**. If you get errors you cannot identify, check the previous function for **unnecessary parentheses**.
- **CandidateList.cpp**
  - Remove the definition of the function **destroyList.**
  - **Destructor:** Leave it empty.
  - **Constructor:** It will be empty, because the vector has its own constructor.
  - Function **addCandidate**: Remove all code and simply use the function **push_back** of the STL vector to insert a new candidate.
  - Function **getWinner**, **printCandidateName**, **printAllCandidates**, **printCandidateDivisionVotes**, **printCandidateTotalVotes,** and **printFinalResults**: Most of the code will stay the same, but you will need to make a few modifications because you are not using a customized linked list any longer. To traverse the **STL vector**, you are not going to use pointers, but you will need to use an **iterator**. If the function is a **const** function, you need to use a **const iterator**.
  - Function **isEmpty:** You do not have a member variable count any longer; how can you check if the vector is empty?

Test your program against the **candidate_list_output** executable file, to make sure everything is working correctly.

SECTION 3 - Adding a private search function

Most of the member functions of the class **CandidateList** perform a search. It is a good idea to create a search function within the class to perform the search every time is needed, and to avoid going through loops if the candidate is not in the list.

Implement the **searchCandidateLocation** function in the **CandidateList** class as follows:

- Make this function a **private** function. (Why? Because it will be used only within the class and it passes an iterator, which is not accessible from outside the class.)
- Returns **true** if the candidate is found or **false** otherwise.

- **Parameters**: A social security number and an **iterator passed by reference** that stores the location where the candidate was found. (Should the iterator be **const**?)
- The function traverses the list using a WHILE loop and stops when the candidate is found or there is no such candidate. **The iterator parameter will indicate where the candidate is located.** Since the iterator is passed by reference, this information will be available to the function that called the **searchCandidateLocation** function.

Now you need to modify the other functions by removing the sections where they are searching for the candidate, and adding instead a **call** to the function **searchCandidateLocation**:

- Function **printCandidateName**
  - Remove all the code and simply add an **IF** statement, which will determine **if** the candidate was found by calling the function **searchCandidateLocation**, and the iterator you are passing in the function call will carry the location **where** the candidate was found. You will use the **iterator** to call the function **printName**.
  - Remove the error message "SSN is not in the list." You will be adding this error message when completing next section.
- Function **printCandidateDivisionVotes**
  - Same as the previous function, but instead of calling the function **printName**, you will obviously call the function **getVotesByDivision**.
  - Remove the error message "SSN is not in the list." You will be adding this error message when completing next section.
- Function **printCandidateTotalVotes**
  - Same as the previous function, but instead of calling the function **printName**, you will obviously call the function **getTotalVotes**.
  - Remove the error message "SSN is not in the list." You will be adding this error message when completing next section.

Test your program to make sure everything is working correctly. You will not have any error message when looking for a candidate that does not exit ➔ **DO NOT FIX IT!** Go instead to the next section.

SECTION 4 - Adding a public search function

This **public** search function will be used by the function **processChoice** to check if the candidate exists, **BEFORE** calling any other functions.

Implement the function **searchCandidate** in the **CandidateList** class as follows:

  - This function is **public**
  - **Parameter**: A social security number.
  - Returns **true** if the candidate is found, and **false** otherwise.
  - The function creates an iterator and calls the function **searchCandidateLocation** to search for the candidate ➔ The function should have two statements only.

Modify the **processChoice** function in the **Main.cpp** file so that **choices 2 and 3**—the ones searching for a specific candidate—check whether the candidate exits by calling the function **searchCandidate**. Make sure you **reason on this**: First you check if the list is empty, and only if the list is empty, you will then check if the candidate is in the list; if the candidate is in the list, then you call all necessary functions to perform the required action, otherwise you will print out the message, "SSN not found." (Note that this is a message to the user, NOT the programmer. What does that mean?)

Test your program against the output_part_2 executable file to make sure everything is working correctly.