Project 2 (Part A): STL Candidate List

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**.

Modify the following:

CandidateList.h	
#include	Include the **STL <vector>** class.
Class **Node**	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**.
destroyList	You have no dynamic variables; therefore, you can delete this function.
isEmpty	Above all the print functions, write the declaration of a function **isEmpty** that returns a **Boolean** value.
printFinalResults	Remove 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	
destroyList	Remove the definition of the function **destroyList.**
Destructor	Since there are no dynamic variables, the destructor will be empty.
Constructor:	It will be empty, because the vector has its own constructor that initializes to an empty vector.
addCandidate	Remove all code and simply use the function **push_back** of the STL vector to insert a new candidate (this is not really efficient, but it simplifies the implementation).
isEmpty	Above all the print functions, write the implementation of the function

	isEmpty that returns true if the vector is empty and false otherwise. Make sure you **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**).
searchCandidate (public)	Modify the function so that it passes an **iterator** in the function call to the private function **searchCandidate.**
searchCandidate (private)	**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 vector 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 this private function.
getWinner **printCandidateName** **printAllCandidates** **printCandidateDivisionVotes** **printCandidateTotalVotes** **printFinalResults**	Remove all errors that indicate whether the list is empty. You will be using the **isEmpty** function from the Main.cpp file to verify whether there are elements in the vector. Most of the code will stay the same, but you will need to make a few syntax 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 **must** use a **const iterator**.
printCandidateName **printCandidateDivisionVotes** **printCandidateTotalVotes**	Do NOT use a loop to search the candidate; use instead the private function searchCandidate to get the location of the candidate whose information needs to be printed.
printFinalResults	Remove this function.
Main.cpp	
processChoice	Modify the **processChoice** function in the **Main.cpp** file so that all choices check if the list is empty before calling any functions. If the list is empty, then you should output the error message, "List is empty." **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.” Which message should be printed as cout and which as a cerr?

TESTING CASES

Compare your output with the given **p2_a_output.exe** file.

- Selection 8
- Selection 1
- Selection 2:
    - 123456789
    - 321452345
    - 111222333
    - 1122
- Selection 3:
    - 123456789
    - 321452345
    - 111222333
    - 1122
- Selection 4
- Selection 5
- Selection 8
- Selection 6
- Try selections 1 and 5 again.
- Comment out the function **createCandidateList** implemented in the **InputHandler.h** file and try the following testing cases (any input will do):
    - Selection 1
    - Selection 2
    - Selection 3
    - Selection 4
    - Selection 5