Lab 9: Vectors

Directions

- You may not work on assignments during the lab, even if you are done the lab.

1. Collecting Phrases

   Write a function with the following prototype:
   ```
   /*
   * Creates a vector of strings by asking the user to enter phrases.
   * Stops when user enters a phrase of length 0 characters.
   *   Returns: vector of phrases (strings)
   */
   vector<string> createPhraseVector();
   ```
   - In the function, first create an empty vector of strings. Hint: `#include` both `vector` and `string`
   - Repeatedly reads in a phrase from the user (using a loop) until the user enters a phrase of zero length (just hits enter).
     - Phrases may have multiple words. Use the following code to read in a full line of text into a string variable named `phrase`:
       ```
       string phrase;
       cout << "Enter a phrase: ";
       getline(cin, phrase);
       ```
       - **Hint:** You can find the length of a string using:
         ```
         string message = "Hello";
         int daLength1 = message.size();
         ```
         Or, use the `empty()` member function:
         ```
         if (message.empty()) {...}
         ```
   - Add each phrase into your vector of strings.
   - When your function finishes (i.e., when the user enters an empty phrase), return the vector of strings which you created.

2. Write a function named `displayPhrases()` which accepts one parameter, the phrases (a vector of strings), and displays them. For example, for the following calling (client) code:
   ```
   vector<string> messages = {"Hello world", "Hi!", "Good bye"};
   displayPhrases(messages);
   ```
   generates the following output:
   ```
   Phrases:
   "Hello world"
   "Hi!"
   "Good bye"
   ```
   Your client code will not hard-code the messages vector: it calls `createPhraseVector()` to create it.
3. Have main call `createPhraseVector()` to get phrases from the user, store the vector in a local variable in `main()`, and then pass it to `displayPhrases()`.
   
   - Test your program with one phrase, two phrases, zero phrases, multi-word phrases, single word phrases, and single character phrases.
   
   - Sample test run:
     
     ```
     Enter a phrase: It's a beautiful day!
     Enter a phrase: Hello world.
     Enter a phrase: I code; therefore I am.
     Enter a phrase:
     Phrases:
       "It's a beautiful day!"
       "Hello world."
       "I code; therefore I am."
     ```

4. **Understanding:**
   
   - How to create, write to, and read from vectors.
   
   - How to pass a vector into a function.

## 2. Finding Phrases with a Vector

Continue the phrases program from above.

1. Write a function which is passed the vector of phrases and returns the index of the shortest phrase:
   ```
   int findShortestIdx(vector<string> data);
   ```
   
   - Have `main()` call this function and have `main()` output both the index of the element (an `int`), and the actual element (the phrase) itself.
   
   - **Hint:** You get the index from the return value of the function call, and using that index you can look-up the phrase in the vector. You must not have any `cout` statements in `findShortestIdx();` the `cout` call must be in `main()` here.
   
   - **Hint:** To find the length of the first element of the vector named `myPhrases` use:
     ```
     int daFirstLength = myPhrases.at(0).size();
     ```
     So, use a loop and check out the length of each string, looking for the shortest.

2. Write a function which returns the index of the longest phrase:
   ```
   int findLongestIdx(vector<string> data);
   ```
   
   - Have `main()` call this function, and have `main()` output both the index of the element (an `int`), and the actual element (the phrase) itself.

3. Write a function which accepts a vector of phrases (strings) and swaps the longest and shortest phrases in the vector.
   
   - The function should change the actual vector you are passing in. How can you pass the vector in so that it can be changed? **Hint:** *It's not pass by value!*
   
   - Use some of the functions you have defined above to simplify your task.
   
   - Have `main()` call this function.
4. Have main re-print all the phrases.
   Hint: you already have this function!

5. Test your program.
   - Test with two elements: they should swap.
   - Test with three elements: the correct two should swap.
   - Test it with one element: it should behave OK.
   - Working with zero elements is the next section!
   - Sample output (input shown in bold-underlined):
     Enter a phrase: It's a beautiful day!
     Enter a phrase: Hello world.
     Enter a phrase: I code; therefore I am.
     Enter a phrase:
     Phrases:
     "It's a beautiful day!"
     "Hello world."
     "I code; therefore I am."
     Shortest idx = 1 = Hello world.
     Longest idx = 2 = I code; therefore I am.
     Phrases:
     "It's a beautiful day!"
     "I code; therefore I am."
     "Hello world."

6. Understanding:
   - How to work with the index of elements in a vector.
   - How to pass a vector to a function so it is changed by the function.

3. Challenges
   - Make the phrases program to work when the user enters no phrases.

4. Skills and Understanding
   You should now be able to answer all the "understanding" questions in the previous sections.
   Complete the following to get credit for the lab:
   - Show the TA the following:
     - Your operational programs which complete all of the above tasks.
     - The TA may ask you to explain any section of the lab, or answer any of the "Understanding" questions.
   - Nothing is to be submitted electronically or in hard-copy for this lab.