CISC4080 Lab1

Goal:

1. Implement bubble sort, selection sort iteratively and recursively.
2. If you use C++, learn to use C++ STL class vector; if you use Python, learn to use list. (click on the links for a tutorial on their usages).

Program Execution:

The following example illustrates how your program should work.

1. The program first reads the length of the vector or list (as 5 in example below), then reads this number of integers, and add them to the vector/list.

2. The program then reads a command (**selection**, **bubble**, **rselection**, **rbubble**), and call the corresponding sorting function to sort the vector/list.

3. The program then display the vector/list after being sorted.

Note that the user input are underscored.

```
$ ./lab1.exe #for C++ code  or
             #$python lab1.py #for python implementation
5
74 25 32 99 24
selection
24 25 32 74 99

[storm: zhang]$ ./lab1.exe
3
17 5 23
bubble
5 17 23
```

Detail Requirements:

1. Please name your program as lab1.cpp or lab1.py depending on the language of choice. The following steps are given for C++, please follow similar structures for Python code.

2. Implement the following function to initialize a vector with user input:

/ initialize a vector */*

vector<int> ReadVector()

/* Read in an integer which gives us the number of integers

Read in those integers one-by-one, and add each into the vector

Return the vector

*/

3. Implement the iterative bubble sort function. Your function needs to follow the given prototype:

/* bubble: iterative bubble sort function
@param: vector a
@post-condition: vector a is arranged into ascending order */
void BubbleSort (vector<int> & a)

4. Implement the iterative selection sort function. Your function needs to follow the given prototype. In your main, call this function if user enter command “selection”.

/ iterative selection sort function*
@param: vector a
*@post-condition: vector a is arranged into ascending order */*
void SelectionSort (vector<int> & a)

5. Implement the recursive bubble sort function. Your function needs to follow the given prototype. In your main, call this function if user enter command “rbubble”.

/ recursive implementation of bubble sort function*
@param: vector a
@param: last, specify the range of vector a to be sorted, i.e., a[0…last] is to be sorted
*@post-condition: vector a[0…last] is arranged into ascending order */*
void RecursiveBubble (vector<int> & a, int last)

6. Implement the recursive selection sort function. Your function needs to follow the given prototype. In your main, call this function if user enter command “rselection”.

/ recursive selection sort function*

@param: vector a

@param: first, specify the range of vector a to be sorted, i.e., a[first…n-1] is to be sorted, where n is the size of vector a

*@post-condition: vector a[first…n-1] is arranged into ascending order */*

void RecursiveSelection (vector<int> & a, int first)

What to submit

Submit your code in C++, named lab1.cpp, by going to the following autograder page:

https://storm.cis.fordham.edu:8443/web/project/1507

For implementation in Python, please submit your program, named lab1.py, to the following page:

https://storm.cis.fordham.edu:8443/web/project/1761

You have a total of 30 submissions opportunities, and 16 submissions on a particular day.

Grading:

Automatic test cases (50 pts): 5 test cases each tests one of the sorting functions. For this to work, pay attention to the following details:

- The command used in your program needs to match with those specified in this description, i.e., selection, rselection, bubble, rbubble
- Before submitting your program, comment out or delete any cout statements except for the one that displaying the vector after sorting function is called.

Handgrading (20 pts): style, comment, and logic…

See the autograder page for this project for details.