San José State University
Department of Computer Engineering

CMPE 180-92

Data Structures and Algorithms in C++

Fall 2016
Instructor: Ron Mak

Assignment #9B

Assigned:	Saturday, October 22
Due:	Friday, October 28 at 11:59 PM
URL:	http://codecheck.it/files/16102208071da9lwcydx1nxltoym83gn621
Canvas:	Assignment 9.b. Vector vs. linked List
Points:	100

Vector vs. linked list

In this assignment, you will compare the performances of the Standard Template Library (STL) vector and the STL linked list. A test program causes the vector and the linked list to undergo an identical set of operations, and it will time how long it takes each data structure to perform the operations. The program prints a table that compares the timings for vectors and lists of various sizes.

Classes `SortedVector` (based on the STL vector) and `SortedList` (based on the STK list) keep their data sorted in smallest to largest order. They have an identical set of member functions whose performances will be tested and timed. You need to complete their class definitions in files **SortedVector.cpp** and **SortedList.cpp**, respectively.

Member function `prepend`

For each data structure, use an iterator to append an integer value at the beginning. You may assume that the data arrives in sorted order, largest to smallest. Therefore, you do not need to sort the data into smallest to largest order.

Member function `append`

For each data structure, append an integer value to the end. You may assume that the data arrives in sorted order, smallest to largest. Therefore, you do not need to sort the data.

Member function `find`

For each data structure, use a constant iterator to search for the given value. Return true if the value is found, else return false.

Member function `get_value`

For each data structure, return the value at the given index `i`. You do not need to check the range of the index.

For the list, use a constant iterator to access the `i`th element. Since you can walk a doubly linked list in both directions starting from its beginning or the end, also use a reverse constant iterator and start from the end whenever that is advantageous.

Member function `clear`

For each data structure, use an iterator to repeatedly erase (remove) the head element until the all the elements are gone.

Sample output

You output should be similar to:

```
        |---Prepend---|  |-----Gets----|  |----Search---|  |----Remove---|  |----Append---|
  Size  Vector     List  Vector     List  Vector     List  Vector     List  Vector     List
   100    0 ms     0 ms    0 ms     0 ms    0 ms     1 ms    0 ms     0 ms    0 ms     0 ms
   500    0 ms     0 ms    0 ms     0 ms    3 ms     4 ms    0 ms     0 ms    0 ms     0 ms
  1000    0 ms     0 ms    0 ms     1 ms    6 ms     9 ms    0 ms     0 ms    0 ms     0 ms
  5000    1 ms     0 ms    0 ms     7 ms   35 ms    54 ms    1 ms     0 ms    0 ms     0 ms
 10000    7 ms     1 ms    0 ms    13 ms   75 ms    88 ms    6 ms     1 ms    0 ms     1 ms
 50000  183 ms     7 ms    0 ms    65 ms  317 ms   428 ms  180 ms     5 ms    1 ms     5 ms
100000  777 ms    16 ms    0 ms   125 ms  611 ms   889 ms  793 ms    11 ms    2 ms    12 ms

Done!
```

Since each run on each machine will produce different timings, CodeCheck will not compare the output. However, you should see similar trends – which data structure is faster for certain operations, how the timings increase as the size increases, etc.

Function parameter

Note that in file **PerformanceTests.cpp**, the overloaded function `elapsed_time` has a function parameter. The function to be timed is passed in.

Rubrics

Criteria	Maximum points
Correct program output (similar to the sample output)	**30**
• `SortedVector`:	• `SortedVector`:
○ prepend	○ 3
○ gets	○ 3
○ search	○ 3
○ remove	○ 3
○ append	○ 3
• `SortedList`:	• `SortedList`:
○ prepend	○ 3
○ gets	○ 3
○ search	○ 3
○ remove	○ 3
○ append	○ 3
Class definitions	**70**
• `SortedVector`:	• `SortedVector`:
○ prepend (use iterator)	○ 5
○ append	○ 5
○ find (use constant iterator)	○ 5
○ gets (use constant iterator)	○ 5
○ get_value	○ 5
○ clear (use iterator)	○ 5
○ append	○ 5
• `SortedList`:	• `SortedList`:
○ prepend (use iterator)	○ 5
○ append	○ 5
○ find (use constant iterator)	○ 5
○ gets (use constant iterator)	○ 5
○ get_value (use constant iterators)	○ 5
○ clear (use iterator)	○ 5
○ append	○ 5

You can submit as many times as necessary to get satisfactory results, and the number of submissions will not affect your score. When you're done with your program, click the "Download" link at the very bottom of the Report screen to download the signed zip file of your solution.

Submit the signed zip file into **Canvas: Assignment 9.b. Vector vs. linked List**.