Lab 7 - IntVector intro

Goals

By the end of this lab you should:

- have a good head start on the IntVector assignment
- have fully tested the constructors and accessor functions
- be confident these functions work correctly

Collaboration policy:

For this lab, collaboration **IS ALLOWED**. You will be working on the first stages of this week's programming assignment. You should begin this class in your public workspace, but do not implement within this workspace any of the remaining functions required for the IntVector assignment. **Copy this IntVector class to your private workspace once you have completed the lab and want to begin working on the IntVector assignment.**

Assignment specs:

You are going to build a smaller, simpler version of the STL vector class that is capable of only storing integers. Appropriately, it will be called IntVector. This class will encapsulate a dynamically allocated array of integers.

You are required to come up with a header file (IntVector.h) that declares the class interface and a separate implementation file (IntVector.cpp) that implements the member functions. You will also create a test harness (main.cpp) that you use to test each function you are currently developing. Never implement more than 1 or 2 member functions without fulling testing them with your own test harness. The final test harness (main.cpp) file should include all of the tests you performed on each member function. Feel free to comment out previous tests while you are working on later member functions. However, the final test harness should test ALL of your member functions thoroughly.

Encapsulated (Private) Data Fields

- sz: unsigned
- cap: unsigned
- data: int *

Public Interface (Public Member Functions)

- IntVector()
- IntVector(unsigned size)
- IntVector(unsigned size, int value)
- unsigned size( ) const
- unsigned capacity( ) const
- bool empty( ) const
- const int & at(unsigned index) const

You will not be implementing iterators. On the most part, though, the above functions should work just like the STL vector class member function with the same name. Please see the following pages to get a better idea of what each function should do: STL Vector.

Constructors and Accessors

Constructors

IntVector( ) - the default constructor

This function should set both the size and capacity of the IntVector to 0 and will **not** allocate any memory to store integers (make sure you do not have a dangling (garbage) pointer).

IntVector(unsigned size)

Sets both the size and capacity of the IntVector to the value of the parameter passed in and dynamically allocates an array of that size as well. (Don't forget to initialize all values in your array to 0.)

IntVector(unsigned size, int value)

Sets both the size and capacity of the IntVector to the value of the parameter passed in and dynamically allocates an array of that size as well. This function should initialize all elements of the array to the value of the 2nd parameter.

Accessors

unsigned size( ) const

This function returns the current size (not the capacity) of the IntVector object, which is the value stored in the sz data field.

unsigned capacity( ) const

This function returns the current capacity (not the size) of the IntVector object, which is the value stored in the cap data field.

bool empty( ) const

Returns whether the vector is empty (the sz field is 0).

const int & at(unsigned index) const

This function will return the value stored in the element at the passed in index position. Your function should cause the program to exit if an index greater than or equal to the size is passed in. Because the index is an unsigned int (rather than just int) the compiler will not allow for a negative value to be passed in.

Compiling and Testing

You can compile the entire program, generating an executable named a.out, with the following command line:

```
    g++ -W -Wall -Werror -ansi -pedantic main.cpp IntVector.cpp
```

But when you want to compile the class seperately, without generating an executable, use the command line:

```
    g++ -W -Wall -Werror -ansi -pedantic -c IntVector.cpp
```

And then to generate the executable once you've already compiled the IntVector class separately, use the command line:

```
    g++ -W -Wall -Werror -ansi -pedantic main.cpp IntVector.o
```

What to Submit

For this assignment you will turn in the following files (case sensitive) to R’Sub (galah.cs.ucr.edu):

- main.cpp (test harness)
- IntVector.h
- IntVector.cpp
