ECS 40 **Program #8 (30 pts, 1 hour)** **Winter 2015**

Due: Monday, March 16th, 11:59 PM using handin to p8 of cs40a.
New concepts: maps, exceptions, and assert
Filenames: main.cpp, SVector.h, SVector.cpp, Makefile, authors.txt

Unlike C++, other programming languages have associative arrays that allow array indices to be strings instead of just integers. You are to create a template class, SVector, that implements a constructor, push_back, [], and pop_back, with STL strings as indices. push_back() takes two parameters, first the index string, and second the value stored. You will store the value in an STL vector, and use an STL map to provide a mapping between the STL string indices and their corresponding integer indices. Since the map will always contain a string and int you should typedef it.

1. main(). Your main() will be quite long because it is the only function in main.cpp.
    - 1.1. All exceptions should be caught and reported in main() relying on printing the value returned by what().
    - 1.2. You must specify the exact exception you are catching, and may not use the elipse (…) in your catch.
    - 1.3. You should declare a pointer to a SVector of type <int> so that you can call its constructor within a try-catch statement and still have scope for the whole main function.
2. Operation Files
    - 2.1. Your program should open and read the file specified by the first command line argument of the program.
    - 2.2. The first line of the file contains an integer that indicates the size with which to construct the SVector. Your program should catch the STL exception thrown by the STL vector class in main(), and report it before terminating.
    - 2.3. Each line of the file starts with a char.
        - 2.3.1. If the char is a 'U', then there is an integer followed by a string on the line with which to call push_back.
        - 2.3.2. If the char is an 'A', then it is followed by a string with which to access the SVector using []. Your program should print out the corresponding integer, or throw an exception if the string is an invalid index.
        - 2.3.3. If the char is an 'O', then it is followed by nothing. Normally, your main() should print nothing, but it should catch the exception if the vector is empty. Your SVector::pop_back() should update the map so the string to int mapping is removed. This is a little tricky, since you have only the int from the vector size() with which to work.
3. Exceptions. Your SVector class should throw four exceptions:
    - 3.1. If the constructor size is invalid, then it will just allow the implicit STL bad_alloc exception to pass up to main().
    - 3.2. If the string index is invalid, then it should explicitly throw an STL range_error exception.
    - 3.3. If the SVector is empty when a pop_back is called because the vector is empty, then it should throw an underflow exception of the STL.
    - 3.4. If the string used for push_back is a duplicate then your class will throw an exception of a class named Whoops that you write in SVector.h that has a constructor, and a what() method.
    - 3.5. Your function definitions should indicate what type of exception they throw.
    - 3.6. assert
        - 3.6.1. You must add assertions that would catch the three explicit SVector exceptions of 3.2, 3.3, and 3.4 before they reach the throw statements.
    - 3.7. Makefile
        - 3.7.1. Your Makefile should have an "all:" rule that compiles both p8.out with the NDEBUG defined using the D option of g++, and p8b.out that compiles without NDEBUG defined so that the asserts will work.
        - 3.7.2. You need not compile a mainB.o file for this assignment, so only two lines are needed to create each executable.
    - 3.8. Further specifications
        - 3.8.1. All implementation code for Whoops and SVector must be in SVector.cpp.
        - 3.8.2. Remember to use const wherever appropriate.

Running the program

```
[ssdavis@lect1 ]$ cat SizeTest.txt
-5
[ssdavis@lect1 ]$ p8.out SizeTest.txt
St9bad_alloc
[ssdavis@lect1 ]$ cat PushText.txt
4
U 45stuff
U 83more stuff
U 92stuff
U 10still going
U 8another
U 6resize
[ssdavis@lect1 private]$ p8.out PushText.txt
Duplicated index: stuff
[ssdavis@lect1 private]$ cat AccessTest.txt
3
U8first
U6second
U3third
U93fourth
Asecond
Afirst
Afifth
Athird
Afourth
[ssdavis@lect1 private]$ p8.out AccessTest.txt
Access value: 6
Access value: 8
Range error: fifth
Access value: 3
Access value: 93
[ssdavis@lect1 private]$ cat PopTest.txt
0
U8a
U3b
O
Ab
O
O
O
[ssdavis@lect1 ]$ p8.out PopTest.txt
Range error: b
Underflow error
Underflow error
[ssdavis@lect1 ]$
[ssdavis@lect1 ]$ p8b.out SizeTest.txt
St9bad_alloc[ssdavis@lect1 private]$ p8b.out PushText.txt
p8b.out: SVector.cpp:44: void SVector<T>::push_back(const std::string&, const T&)
[with T = int]: Assertion `indiceMap.find(s) == indiceMap.end()' failed.
Abort (core dumped)
[ssdavis@lect1 ]$ p8b.out AccessTest.txt
Access value: 6
Access value: 8
p8b.out: SVector.cpp:58: const T& SVector<T>::operator[](const std::string&) const
[with T = int]: Assertion `itr != indiceMap.end()' failed.
Abort (core dumped)
[ssdavis@lect1 ]$ p8b.out PopTest.txt
p8b.out: SVector.cpp:58: const T& SVector<T>::operator[](const std::string&) const
[with T = int]: Assertion `itr != indiceMap.end()' failed.
Abort (core dumped)
[ssdavis@lect1 ]$
```