Sonic Boom!

Due Tuesday, April 22 at 8 a.m.

CSE 1325 - Spring 2025 - Homework #11 - Rev 0

Assignment Overview

To wrap up the semester, we'll return to the Boom! assignment and reimplement / enhance parts of it using *different* C++ features. You may baseline your own P09 full_credit, bonus, or extreme_bonus solution for this assignment, or use the extreme_bonus suggested solution (in which case, see the Canvas Announcement "Baselining the Next Sprint on a Suggested Solution" for proper attribution).

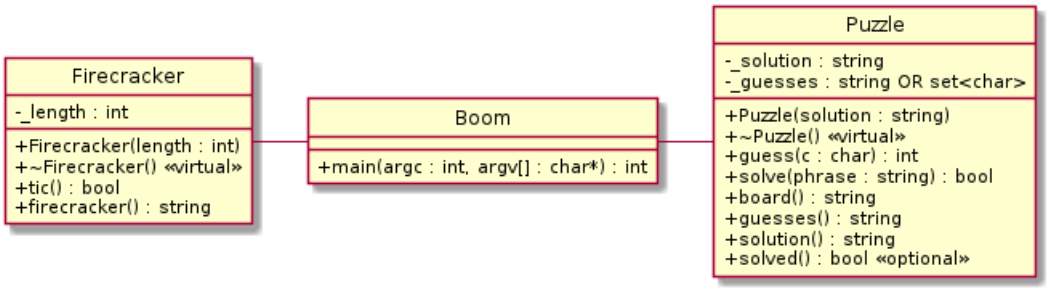
Then make the following changes:

- The first program argument is now the name of a file of quotes, one per line, for puzzles (find quotes.txt and puzzles.txt at cse1325-prof/P11/baseline). Load them all into a `std::set`, then select one at random to instance the `Puzzle` *using an iterator*.
- In the `Puzzle` constructor, force all chars in the solution to lowercase. Hint: `tolower` is the go-to function using a for-each reference loop. (You'd think we'd have a `std::string` method instead of looping through and updating every `char`, but we don't.)
- Rewrite `Puzzle::guess` to return the number of times the c appears in `_solution`. Use `std::count`. Print this number as part of your user interface in the main function in file boom.cpp.
- Rewrite `Puzzle::board` *without using any strings*. Instead, use an input string stream (from field `_solution`) and an output string stream to create the board. Show all chars that are not `std::isalpha` or have been added to `_guesses`, otherwise show an '_'.

Full Credit

Consider the following slightly modified class diagram from Assignment #9, which implements a word guessing game in which the Firecracker burns one segment for each letter guessed and missed, and the firecracker goes “Boom!” if the Firecracker is expended before the phrase is guessed. The player may guess the solution at any time, winning if correct or immediately getting a “Boom!” if incorrect.

Detailed guidance is provided starting on the next page.



Loading a Random Puzzle

Several text files containing one puzzle per line are provided at cse1325-prof/P11/baseline.

- File **quotes.txt** contains long inspirational puzzles containing only lower case letters and spaces that are easy to solve.
- File **puzzles.txt** contains shorter, more challenging puzzles from the TV game show "Wheel of Fortune" with upper case letters and punctuation marks. Using this file will require that `Puzzle::board` show all non-`std::isalpha` chars as described below.

Modify the main function in file boom.cpp to select a random puzzle from the file specified by the first command line argument.

- Open the file using a `std::ifstream`.
- Create a `std::set` (NOT an array or vector of strings!) to hold the quotes from the file.
- Use a loop with `std::getline` to load ALL of the lines from the file into the set.
- Generate an `index` to a random quote in the set. Hint: `srand(time(NULL));` at the beginning of main will "seed" the default pseudo-random number generator, and `srand()` will return a random non-negative int.
- Since `std::set` is not sequenced and thus accepts no index, obtain an iterator pointing to the first quote in the set and iterate forward `index` times so that it is pointing to the selected quote.
- Instance the `Puzzle` using the (dereferenced) selected quote.

Forcing the Solution to Upper Case

C++ lacks a `std::string` method or function to force all characters in a string to upper or lower case. It does have the `toupper` and `tolower` functions from C, however.

A for-each loop with a reference variable can modify each affected char in place in the `_solution`, though. Nice that `std::string` is mutable, isn't it?

Return the Number of Guessed Char in Solution

`std::count` counts the number of times the third parameter appears in the collection (what C++ calls a container). For maximum flexibility, the first two parameters are iterators pointing to the first element and one past the last element to count.

For example, given `std::vector<std::string> names`, we could count the number of times "Prof Rice" appears in the entire vector using

```
int n = std::count(names.begin(), names.end(), "Prof Rice");
```

If we wanted to search just the most recent 10 entries, try

```
int n = std::count(names.end()-10, names.end(), "Prof Rice");
```

Or the first ten entries,

```
int n = std::count(names.begin(), names.begin()+10, "Prof Rice");
```

See the power of C++ iterators when used with the library?

Now apply this to count all of the times the guess appears in the `_solution` string.

Creating the Board String with No Strings

We'd normally use a string here (you probably did!). let's re-implement without strings!

First, instance an output string stream (`std::ostringstream`) into which we can stream characters for the board. Streaming a `char` into a stream will likely require a *proper* cast (NOT just parentheses - see Lecture 22) - otherwise, C++ will annoyingly convert the `char` to an `int`.

Next, instance an input string stream (`std::istringstream` streaming in from `_solution`) so that we can stream each `char` out one by one.

Using the >> operator to stream out each `char` will skip the spaces, and those are important! So we'll use stream's `get(c)` method to read each `char` into `char c` - *including* each space.

Now loop through each `char` in the input stream, streaming to the output stream the transformed `char`. That is, if the `char` has been guessed or isn't `std::isalpha`, stream it as is, otherwise stream '_' instead.

Finally, return all of the text streamed to the output string stream.

Running Boom!

Boom's singl argument is now the name of a file in which each line contains a possible puzzle to be guessed. Two are provided at cse1325-prof/P11/baseline, one with long and easy inspirational quotes and another with actual puzzles from "Wheel of Fortune". The latter requires ALL of the above changes, since they are in capitals and include punctuation which must be shown as-is and NOT count against the player.

So run the program like this:

```
./boom quotes.txt

./boom puzzles.txt
```

Makefile

The Makefile doesn't change from P09.

```
CXXFLAGS = --std=c++20

boom: boom.o puzzle.o firecracker.o
	-$(CXX) $(CXXFLAGS) boom.o puzzle.o firecracker.o -o boom
	@printf "Now type './boom <file of puzzles>' to play the game!\n\n"

firecracker.o: firecracker.cpp *.h
	-$(CXX) $(CXXFLAGS) -c firecracker.cpp -o firecracker.o

boom.o: boom.cpp *.h
	-$(CXX) $(CXXFLAGS) -c boom.cpp -o boom.o

puzzle.o: puzzle.cpp *.h
	-$(CXX) $(CXXFLAGS) -c puzzle.cpp -o puzzle.o

clean:
	rm -f *.o *.gch a.out  boom
```

Example Output

```
ricegf@antares:~/dev/202501/P11/full_credit$ make
g++ --std=c++20 -c boom.cpp -o boom.o
g++ --std=c++20 -c puzzle.cpp -o puzzle.o
g++ --std=c++20 -c firecracker.cpp -o firecracker.o
g++ --std=c++20 boom.o puzzle.o firecracker.o -o boom
Now type './boom <file of puzzles>' to play the game!

ricegf@antares:~/dev/202501/P11/full_credit$ ./boom puzzles.txt
12881 quotes loaded.
d
         =================
             B O O M !
         =================

Enter lower case letters to guess,
! to propose a solution,
0 to exit.


  ------
 | BOOM |-------*
  ------


Guessed:
___ ___ ____ ______ _________: s

  ------
 | BOOM |-------*
  ------

Found 2 s

Guessed:  s
___ ___ ____ ______ _____ss__: m

  ------
 | BOOM |------*
  ------

Found 0 m

Guessed:  ms
___ ___ ____ ______ _____ss__:
```

Bonus

Bonus work will be provided if time permits.