Examination in Programming Proficiency (EPP) Part II

- The exam is **open textbook, open notes.**
- **Internet access is allowed to look up reference material but not for copy&paste of any existing solution (e.g., from chegg.com, coursehero.com)**. This will be considered academic dishonesty
- Collaboration with anyone is not allowed (this will be considered academic dishonesty).
- You might find yourself under some time pressure in this examination. Please check the point value for each problem so that you do not spend more time on one problem than it is worth.
- Please make sure to read each problem carefully before working on it.

IMPORTANT:

- **Starter code is given to you on Titanium**
- Upload ONLY your C++ files (.cpp, .h) to Titanium.

Total points: 40

Approximate grading guidelines:

5 points for design and documentation, 35 points for the actual code implementation.

Complete the given program (part2.cpp) to calculate the hitbox size of PC game characters. A hitbox is an invisible rectangle surrounding a character and is used to detect collisions. Each character has a length and width (in centimeters) and the product of this length and width is the hitbox size (in square centimeters). For instance, if a character is 10cm long and 5cm wide in relation to the map, the hitbox size for the character is 50 sq. cm.

Implement class Hitboxes (in file Hitboxes.h) which should include the following public member functions:

- `Hitboxes(string filename);`

The constructor takes in a filename and reads in the data to its member variables. Each line in the text file contains four pieces of information separated by whitespaces: character's name (one string), character's type (one string), length (int), and width (int). Note: file reading should take place only during object construction - not in any other method.

A sample file is shown below:

```
Bloodhound Scout 8 5
Pathfinder Scout 10 6
Lifeline Defense 8 5
Gibraltar Defense 10 8
Mirage Soldier 11 4
Bangalore Soldier 11 4
Wraith Soldier 8 4
Caustic Defense 13 5
```

- `string smallestCharacter();`

returns the name of the character with the smallest hitbox size. For the sample file shown above, smallestCharacter() should return `Wraith`.

- `string smallestType();`

returns the type whose characters taken together have the smallest total hitbox size. For the sample file shown above, `smallestType()` should return `Scout`.

The given `part2.cpp` calls these methods and tests the results. All of your code should go in Hitboxes.h. You should add member variables, functions, classes/structs, and libraries to your code as needed. You can modify the main function to test your code with different input cases to make sure the selection logic is correctly implemented.

- You can implement all your code either in one header file called `Hitboxes.h` or split the declaration and definition in `Hitboxes.h` and `Hitboxes.cpp`

Simplifying assumptions/hints:
- You do not need to do any error-checking
- Since the names do not contain spaces, you can read each line using code like:
`myfilestream >> myname >> mytype >> myint1 >> myint2;`

Data structures: **you are encouraged to use the C++ Standard Library containers.**

Required documentation:

- Design of your class. This should be turned in as a long comment at the beginning of the Hitboxes.h. Make clear what, if any, data structures you use and their roles.
- Comments:
  - Any unusual or tricky code should be commented, but do not just repeat the code.

File submission: Upload as separate files (do not zip):

- `Hitboxes.h`
- `Hitboxes.cpp and other files, if applicable`

COMMAND TO COMPILE AND RUN CODE ON LINUX/TUFFIX (RECOMMENDED):

```
clang++ -std=c++17 part2.cpp

./a.out
```