# Exam in Programming Proficiency Part II

Directions:

Published text will be allowed (open book policy), but no lecture notes, copies of personal program listings, or electronic media (hard drives, flash drives, cell phones, etc. ) will be allowed except for authorized software (compiler, word processor, IDEs like Visual Studio, XCode) already installed on the lab computers. Furthermore, no access to the Web will be allowed during the exams except for accessing the Titanium site.

Please PRINT your name: _____

Please SIGN your name: _____

**IMPORTANT:**

- **Starter code and data files are given to you on Titanium. The last page has instructions on how to get these into Linux and Visual Studio**
- **Upload ONLY your C++ files (.cpp, .h) to Titanium. Do NOT upload any Visual Studio solution files.**

Total points: 40

Approximate grading guidelines:

10 points for design and documentation, 30 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 structure resources**:

**You are encouraged to use the C++ Standard Library containers.**

However, you are also given files containing templates for a Stack class, a Queue class, and a Vector class. For each of the three data structures, you are also provided with a test program**.** You may use none, any, or all of them if it will help you. You are also free to edit any of these files. These files are posted in the "EPP Part 2 data structure implementation files" section.

**Required documentation:**

- Design of your class. This should be turned in as a long comment at the beginning of the main program file. Make clear what, if any, data structures you use and their roles.
- Comments:
  - All functions, aside from the main, must have a brief documentation that states what the function does, and the roles played by each of the parameters, if any, as long as the name of the function and the names for the parameters do not make it clear.
  - 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`
- `Any other data structure files that you used`

**COMMAND TO COMPILE AND RUN CODE ON LINUX/TUFFIX:**

```
clang++ -std=c++17 part2.cpp
./a.out
```

## How to get starter code and data files into Visual Studio
**(and ensure that you do not have multiple versions of your code on the PC)**

1. **Download** prob2.cpp and the .txt files from the Titanium Part 2 assignment link. Some other data structure code (vector, stack, queue) is also given to you.

2. Create an **Empty Project** in Visual Studio. Remember where your new project is created (something like `C:\Users\myname\Documents\Visual Studio 2015\Projects\MyProject2\MyProject2`). Do not add any new items to your project yet.

3. **Move** the downloaded .cpp and .txt to the Visual Studio projects folder (i.e., from Downloads to `C:\Users\myname\Documents\Visual Studio 2015\Projects\MyProject2\MyProject2`).

4. In Visual Studio, do `Project` → **`Add Existing Item`** → select part2.cpp (and any other downloaded .h code)

5. In Visual Studio, do `Project` → **`Add New Item`** → `Header file (.h)` → type BirthAnalysis.h. Do this for other new .cpp/.h files that you create

6. At the end of the exam, **submit to Titanium** the .cpp and .h files from the Visual Studio projects folder