PS4: Airport Simulation Project (C++11 Concurrency)

In this assignment you will complete a program that simulates landings of multiple airplanes on multiple runways at Logan Airport. See the accompanying slides that describe the airport layout, and rules regarding landings in this simulation.

Download the project starter code (a Zip file) from Blackboard. When you extract the starter code, it will be in a directory named `Airport-Nosync`. This directory contains a working implementation of the project, but **without** the necessary mutual exclusion and synchronization needed to prevent airplane crashes due to violations of landing rules (e.g., improper simultaneous use of runways). Using the included `Makefile`, build and execute the provided solution to make sure that your initial environment is set up correctly.

When satisfied that the initial implementation is working correctly on your system, rename the base directory as `Airport`. Now, make the following changes to files in your `Airport` directory:

1. Modify the file `Airport.cpp` such that the thread constructor call on line 34 uses a lambda expression as its parameter, rather than a function pointer. (Test your solution by re-making the executable – the project should run as before, albeit still with crashes.)
2. Modify the file `AirportServer.h` to include declarations of the mutex variables and condition variable that you will need in `AirportServer.cpp`, as indicated by the comments in this file.
3. Modify the file `AirportServer.cpp` to include mutex operations and condition variable operations as necessary to enforce the rules of the simulation, as indicated by the comments in this file. When your solution is correct, it should:
    a. run without “crashes” (run it for at least 15 minutes!);
    b. allow airplanes to land simultaneously on non-conflicting runways as allowed by the rules of the simulation; and
    c. allow up to (but no more than) six simultaneous landing requests to Air Traffic Control.

    For reference, the included (Zipped) output.txt file contains sample output from a 15+ minute run of a completed solution. (Note that due to concurrency and random number generation used in the simulation, your output should be similar, but not identical!)
4. Complete the file `Airport-readme.txt`.

Note: You should not need to modify any files other than the files listed above!

What to turn in

Create a tarball of your `Airport` directory structure. Include your name in the archive file name (e.g., `Tom_Wilkes_PS4_Airport.tar.gz`)

How to turn it in

Submit your tarball via the PS4 assignment page on Blackboard.

Grading rubric

Feature	Value	Comment
Airport implementation	**18**	**full & correct implementation** 2 pts uses lambda expression correctly in `Airport.cpp` 5 pts runs “forever” without the airport closing due to plane crash (at least 15 minutes) 3 pts declares mutex variables and a condition variable correctly in `AirportServer.h` 5 pts uses mutex variables correctly in `AirportServer.cpp` to provide mutual exclusion for landings on runways as needed 3 pts uses a condition variable correctly in `AirportServer.cpp` to allow simultaneous active landing requests from up to (but no more than) six Airplanes
tar.gz archive	**1**	**all files packaged in .tar.gz file with correct directory structure**
Airport-readme.txt	**3**	**complete and discusses work**
Total	**22**	