```
    1: #include "AirportRunways.hpp"
    2:
    3: int AirportRunways::runwayInUse[AirportRunways::NUM_RUNWAYS];
    4:
    5: int AirportRunways::numLandingRequests = 0;
    6:
    7: int AirportRunways::maxNumLandingRequests = 0;
    8:
    9: mutex AirportRunways::checkMutex;
   10:
   11:
   12: string AirportRunways::runwayName(RunwayNumber rn)
   13: {
   14:         switch (rn)
   15:         {
   16:         case RUNWAY_4L:
   17:                 return "4L";
   18:         case RUNWAY_4R:
   19:                 return "4R";
   20:         case RUNWAY_9:
   21:                 return "9";
   22:         case RUNWAY_14:
   23:                 return "14";
   24:         case RUNWAY_15L:
   25:                 return "15L";
   26:         case RUNWAY_15R:
   27:                 return "15R";
   28:         default:
   29:                 return "Unknown runway " + rn;
   30:         } // end switch
   31:
   32: } // end AirportRunways::runwayName()
   33:
   34:
   35:  /**
   36:   *  Check the status of the aiport with respect to any violation of the rul
es.
   37:   */
   38: void AirportRunways::checkAirportStatus(RunwayNumber requestedRunway)
   39: {
   40:         lock_guard<mutex> checkLock(checkMutex);
   41:
   42:         bool crash = false; // Set to true if any rule is violated
   43:
   44:         cout << "\nChecking airport status for requested Runway " << runwayN
ame(requestedRunway) << "..." << endl;
   45:
   46:         requestRunway(requestedRunway);
   47:
   48:         // Check the number of landing requests
   49:         cout << "Number of simultaneous landing requests == " << numLandingR
equests
   50:                  << ", max == " << maxNumLandingRequests << endl;
   51:
   52:         if (numLandingRequests > MAX_LANDING_REQUESTS)
   53:         {
   54:                 cout << "***** The number of simultaneous landing requests e
xceeds Air Traffic Control limit of " << MAX_LANDING_REQUESTS << "!\n";
   55:                 crash = true;
   56:         }
   57:
```

```
   58:         // Check the occupancy of each runway
   59:         for (int i = RUNWAY_4L; i <= RUNWAY_15R; i++)
   60:         {
   61:                 cout << "Number of planes landing on runway " << runwayName(
RunwayNumber(i)) << " == " << runwayInUse[i] << endl;
   62:
   63:                 if (runwayInUse[i] > 1)
   64:                 {
   65:                         cout << "***** The number of planes landing on runwa
y " << runwayName(RunwayNumber(i)) << " is greater than 1!\n";
   66:                         crash = true;
   67:                 }
   68:         }
   69:
   70:         // Check individual restrictions on each runway
   71:         if ((runwayInUse[RUNWAY_9] > 0)
   72:                 && ((runwayInUse[RUNWAY_4R] > 0) || (runwayInUse[RUNWAY_15R]
 > 0)))
   73:         {
   74:                 cout << "***** Runways 9, 4R, and/or 15R may not be used sim
ultaneously!\n";
   75:                 crash = true;
   76:         }
   77:
   78:         if (((runwayInUse[RUNWAY_15L] > 0) || (runwayInUse[RUNWAY_15R] > 0))
   79:                 && ((runwayInUse[RUNWAY_4L] > 0) || (runwayInUse[RUNWAY_4R]
> 0)))
   80:         {
   81:                 cout << "***** Runways 15L or 15R may not be used simultaneo
usly with Runways 4L or 4R!\n";
   82:                 crash = true;
   83:         }
   84:
   85:         // If any of the rules have been violated, terminate the simulation
   86:         if (crash)
   87:         {
   88:                 cout << "***** CRASH! One or more rules have been violated.
Due to the crash, the airport is closed!\n";
   89:                 exit(-1); // Abnormal program termination
   90:         }
   91:
   92:         // Status check is normal
   93:         cout << "Status check complete, no rule violations (yay!)\n";
   94:
   95: } // end AirportRunways::checkAirportStatus()
```