Introduction

Consider a coach with capacity of 36 passengers. The seats of the coach are arranged in 9 rows and 4 columns (i.e., 4 seats on each row). Each passenger is allowed to stow a piece of luggage in the luggage compartment. A luggage record sheet, in the form of a seat-layout plan (9 * 4 matrix), is used to keep this record.

Tasks

Using Pthreads, design and implement a multithreaded C++ program under Linux that works as follows:

- A master thread is created to model the driver of the coach.
  - The driver has to assign a seat number to each passenger in the form of [row, column] where 1 ≤ row ≤ 9 and 1 ≤ column ≤ 4 (Hint 1).
  - The driver has to wait until all the passengers (**N**≤36) are on board.
  - Before departure, the driver has to check the luggage record sheet to determine the number of passengers stowing luggage in the compartment (Hint 4).
- Each passenger is modeled with a worker thread (created by the master thread).
  - Each passenger is assigned a seat number (Hint 1).
  - Passengers get on the coach at random times, ranging from 0 to 10 minutes (Hint 2).
  - Every on-board passenger has to mark the corresponding entry on the luggage record sheet, according to his/her seat number, to indicate whether he/she has stowed a luggage in the compartment. Suppose, on average, 20% of the passengers carry a luggage (Hint 3).

Hints

1. **Passing parameters to each thread**: When the master thread (driver) creates the N worker threads (passengers), it passes to each worker thread a seat number in row-major order ((1, 1), (1, 2), (1, 3), (1, 4), (2, 1), … , (9, 4)). This step requires passing two numbers (row and column) to each worker thread.
2. **Simulating random arrivals**: One way to make passengers get on the coach at random times is to have the worker threads (passengers) sleep for a random period of time upon creation. For example, `sleep(rand()%(10*60+1))` (http://linux.die.net/man/3/sleep, http://www.tutorialspoint.com/cplusplus/cpp_numbers.htm) can be used to simulate the random arrival times of passengers, ranging from 0 to 10 minutes.
3. **Simulating random luggage-carrying passengers**: One simple way to randomly choose approximately 20% of the passengers to carry a luggage is to check `rand()%101 <= 20` in each worker thread. If yes, the passenger carries a luggage; otherwise, there is no luggage.
4. **Returning results to the master thread**: A luggage record sheet is used to show which passengers have stowed luggage in the compartment. One way to handle this is to create a matrix (record sheet) that is accessible to all threads. If a worker thread sets its corresponding entry to 'Y', it indicates that there is a luggage; otherwise, the entry is set to 'N'. An entry of 'E' would indicate an empty seat. When all passengers are on board the coach, the master thread counts the number of entries with 'Y' in the matrix to determine the number of passengers stowing luggage in the compartment.
5. **Passing command-line arguments to your program**: To pass command-line arguments (which are character arrays) into your program and convert the arguments into numbers, read http://www.site.uottawa.ca/~lucia/courses/2131-05/labs/Lab3/CommandLineArguments.html.

Program requirements and marking scheme

- Design and use of multithreading (**60%**)
  - Complete, correct and thread-safe multithreaded design and implementation including
    - creation and termination of master thread
    - creation and termination of worker threads
    - passing parameters to worker threads
    - returning results to master thread
  - Non-multithreaded implementation (0%)
- Program correctness (**30%**)
  - Complete and correct implementation of other features including
    - correct logic and coding of thread functions
    - passing N (number of passengers) to the program on the command line
    - random arrivals of passengers
    - approximately 20% of passengers carry luggage
    - program input and output conform to the format of the sample below
    - successful program termination
  - Fail to pass the g++ complier on our course Linux server cs3103-01.cs.cityu.edu.hk to generate a runnable executable file (0%)
- Programming style and documentation (**10%**)
  - Good programming style
  - Clear comments in the program to describe the design and logic (no need to submit a separate file for documentation)
  - Program completed in one file
  - Unreadable program without any comment (0%)

Input / Output sample

```
>coach 25
NNNN
NYNN
YNNN
NYNN
NNYN
NNNY
NEEE
EEEE
EEEE
5 passengers carry luggage.
```

Submission

- This assignment can be done individually or in a group of two members. The grouping must be the SAME for ALL the programming assignments in this course.
- Each submission consists of two files: a source program file (.cpp file) and a text file containing TWO different input/output samples generated by your program (.txt file).
- Use your student ID(s) to name your submitted files, such as 5xxxxxxx.cpp, 5xxxxxxx.txt for individual submission, or 5xxxxxxx_5yyyyyyy.cpp, 5xxxxxxx_5yyyyyyy.txt for group submission. Only ONE submission is required for each group.
- Submit the files to Canvas.
- The deadline is **10:00 a.m.**, **22-FEB-16 (Monday)**. No late submission will be accepted.