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. ***The coach is running in a circular route to carry passengers from the terminal to the airport.***

Tasks

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

- Each passenger is modeled with a worker thread.
  - Passengers arrive at random times, ranging from 0 to 10 minutes (600 seconds).
  - On arrival, every passenger gets a seat number from an automatic ticketing machine[1] which issues a seat number in the form of (row, column), where 1 ≤ row ≤ 9 and 1 ≤ column ≤ 4, in row-major order, i.e., (1, 1), (1, 2), (1, 3), (1, 4), (2, 1), … , (9, 4).
  - ***If the coach has not arrived, passengers wait in the terminal.*** When the coach arrives, each passenger gets on the coach after a random time, ranging from 0 to 1 minute (60 seconds).
  - Every on-board passenger writes down his/her seat number and indicate whether he/she has stowed a luggage in the compartment on a luggage tag and put it into a collection box[2] in order. Suppose, on average, 20% of the passengers carry a luggage.

Luggage Tag

Seat No.: (,)

Luggage: Y / N

- ***The driver of coach (or simply the coach) is modeled with a worker thread.***
  - ***At time 0, the coach is in the terminal. It departs the terminal to the airport when it becomes full^ and it takes 10 minutes from the last departure for the coach to return to the terminal for the next trip.***
    ^ You can assume that the total number of passengers (N) is a multiple of 36 in this assignment.
  - On departure, the driver reads the luggage tag in order and marks the corresponding entry on the luggage record sheet according to the seat number as follows and determines the number of passengers stowing luggage in the compartment.
    - An entry of ‘Y’ indicates that the corresponding passenger has stowed a luggage in the compartment.
    - An entry of ‘N’ means the corresponding passenger does not carry any luggage.
    - ~~An entry of ‘E’ indicates an empty seat.~~ (The coach is always full on departure.)
  - ***After reading all the luggage tags, the driver clears the luggage tag collection box.***
- ***The main function is the master thread***.
  - It creates all the worker threads (the driver and passengers) in the beginning of the program.
  - ***The program terminates when all the passengers have departed the terminal.***

Detailed requirements

1. **Automatic ticketing machine**
    - Write a function to simulate the automatic ticketing machine which issues seat numbers in row-major order. Specifically, the function needs to keep track of the row number and column number of the last issued seat number. When a passenger calls the function to get the next seat number, it takes one second (by using sleep(1)) to generate the next row number and then takes another one second to generate the next column number.
    - ***The machine starts operation at time 0. It stops operation after the last seat number (9, 4) is issued. When the coach departs the terminal, the machine resets and restarts the operation for the next trip.***
2. **Luggage tag collection box**
    - Write two functions, enqueue and dequeue, to simulate the two operations on the collection box, which is a queue-like data structure*. Every on-board passenger calls enqueue to insert a luggage tag into the queue in FIFO (first-in-first-out) order and the driver calls dequeue to read a luggage tag in FIFO order.
      * It is acceptable to use an array to model the queue-like data structure.

Program requirements and marking scheme

- Design and use of multithreading, mutex and ***semaphore*** (**60%**)
    - Complete and correct design and implementation of
        - thread management
        - mutual exclusion
        - ***thread synchronization***
    - ***No deadlock and busy waiting***
    - Non-multithreaded implementation without considering mutual exclusion and thread synchronization (0%)
- Program correctness (**30%**)
    - Complete and correct implementation of other features including
        - correct logic and coding of all (thread and non-thread) functions
        - 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 samples[#]

```
>coach 36                          >coach 72
Coach arrives at time 0.           Coach arrives at time 0.
Coach departs at time 210.         Coach departs at time 321.
NNNN                               NYYN
NYNN                               NNNN
YNNN                               YNNN
NYNN                               NYNN
NNYN                               NNYN
NNNY                               NNNY
NNNN                               NNNN
YNNN                               YNNN
NNNY                               NNYN
7 passengers carry luggage.        8 passengers carry luggage.

                                   Coach arrives at time 921.
                                   Coach departs at time 975.
                                   NNNN
                                   NNNN
                                   NNYY
                                   NNNN
                                   NNYN
                                   NNNY
                                   NNNN
                                   NNNN
                                   NNNY
                                   5 passengers carry luggage.
```

[#] The time is in seconds. To calculate the time difference in seconds, you need to use certain time functions such as `time` and `difftime` (http://www.tutorialspoint.com/cplusplus/cpp_date_time.htm).

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*** input/output samples (***N=36 and N=72***) 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., 26-APR-16 (Tuesday)**. No late submission will be accepted.