EECS 111 System Software Spring 2023
Project 2: Thread & Synchronization

Due Date (May 25th, 2023 11:55 PM)

Instructions

In this project, you will work on thread synchronization using mutex locks to solve a proble m. The first step is to read and understand this project description. Then, you can start coding. By unzi pping the project file you will see the following files in the ***p2_student*** directory:

1. `main.cpp`: This is a top file for this project
2. `p2_threads.h`: This is a header file to handle threads
3. `p2_threads.cpp`: This is a source file to handle threads
4. `utils.h`: This is a header file for utility functions
5. `utils.cpp`: This is a source file for utility functions
6. `types_p2.h`: This is a header file for the Person class
7. `types_p2.cpp`: This is a source file for the Person class
8. `Makefile`: This is a compilation script

You need to complete the source files: `main.cpp`, `p2_threads.cpp`, and `types_p2.cpp`. You should be able to build the project code by the command `make`. After the build, **you must have `p2_exec` as the file name of your executable.** Also, it is suggested to remove all the compiled obje ct files and executable files with the command: `make clean`. You can change the source code in the given files. In addition, you can add any new header and source code files for this assignment. But, pl ease make sure that your program can be compiled with the `make` command. The executable will take only 1 argument. You have to print error messages and usage examples if you didn't provide the argu ment. The following is an example error message.

```
[ERROR] Expected 1 argument, but got (X).
[USAGE] p2_exec <number>
```

Problem Description

Suppose that a university wants to show off how progressive it is and ends its long-standing practic e of gender-segregated restrooms on campus. However, as a concession to propriety, it makes a policy that when a woman is in the restroom only other women may enter, but not men, and vice versa. On t he door of every restroom, there will be a sign with a sliding marker that will indicate one of three pos sible states it is currently in:

`Empty`, `WomenPresent`, and `MenPresent`

For this assignment, you will need to complete a working program that will address the above prob lem and that must compile and run. The program must execute until all the people finish using the rest room.

- The program must contain the following functions: `woman_wants_to_enter`, `man_wants_to_enter`, `woman_leaves`, and `man_leaves`.
- The program must display the following during its execution:
  - The timestamp
  - The state of the restroom (empty, occupied by women and if so how many, occupied by men, and if so how many)
  - The status of the queue including whether it is empty or not
  - If not empty what genders are in the queue

Format:
```
[<Time> ms][<Thread name>] <Behavior>, <Thread status>: Total: <numb
er> (Men: <number>, Women: <number>)
```

The example command line output for each case is as follows:
```
[12 ms][Queue] Send (Man) into the restroom (Stay 4 ms), Status: Tot
al: x (Men: x, Women: x)
[12 ms][Restroom] (Man) goes into the restroom, State is (MenPresen
t): Total: x (Men: x, Women: x)
[16 ms][Restroom] (Man) left the restroom. Status is changed, Status
 is (empty): Total: x (Men: x, Women: x)
```

- The above example output is only for one possible scenario. There may be many additional command line outputs.
- During run-time, you need to generate a sequence of people based on the input argument. If 10 is the input argument, then your program must generate a sequence of a total of 20 people that includes 10 men and 10 women.
- You must randomly assign the gender to a person and send that person to the restroom. After that person has been sent to the restroom, you must wait for a random time interval (between 1 milliseconds – 5 milliseconds) until you can send the next person to the restroom. Your program must display the following output:

```
[01 ms][Input] A person (Man) goes into the queue
[03 ms][Input] A person (Woman) goes into the queue
```

- When a person goes into the restroom, you must randomly assign the time to stay in the restroom (3 milliseconds – 10 milliseconds). Your program must display the following output:

```
[56 ms][Queue] Send (Man) into the restroom (Stay 4 ms), Status: Tot
al: x (Men: x, Women: x)
[69 ms][Queue] Send (Woman) into the restroom (Stay 7 ms), Status: T
otal: x (Men: x, Women: x)
```

- Your program must generate a different sequence each time it is running. In other words, your program must show different behaviors for each run of the program.

- Your code MUST SHOW parallel behavior
- Your program should not have a starvation problem
- TIP: You must have at least two threads, not including the main process

Submission

You need to compress all the files into a single archive **.zip file** (no other format will be accepted) and upload it. The deadline for uploading the files is the project deadline. Since your submissions will be processed by a program, there are some very important things you must do, as well as things you must not do:

1. It is imperative to keep the output format the **EXACT** same as what you are asked for.
2. You MUST USE only the C++98 standard. If you use any new features like C++11 or C++1 4, your code will not be graded.
3. Make sure your code can be compiled and run. If we cannot compile your code, then your cod e will not be tested and graded.
4. Your executable filename must be **`p2_exec`**.
5. Your code **MUST SHOW** a parallel behavior. Otherwise, your code will not be graded prope rly.
6. Your submissions must contain all your source code and Makefile in the **root** of the zip file.
7. When zipping your source code, use the following command in a Linux/Unix terminal while you are inside the directory with your source code:
    a. zip -j p2_<studentId>.zip <list of your source code files>
    b. Example for above:
    `zip -j p2_1234.zip main.cpp p2_threads.h p2_threads.cpp …`
8. The codes that the students submit will be compared with each other using a program. If they are similar enough, the program will give us a warning about it. So, you can talk to each other about the project, and visit online resources, but you must write your own code.

Grading

1. (5%) Following the submission format
2. (10%) Compile your code with your Makefile without any problem.
3. (10%) Command-line output matches with the description for any type of input.
4. (15%) Randomly generate the input sequence and randomly assign the time to stay in the restro om
5. (40%) Your program works properly and does not have a deadlock.
6. (20%) Your program does not have a starvation problem.

Note

- Even though you can generate correct output, that does not mean that your code has considered extreme cases. You should verify your code with some corner cases as well.
- If you have any questions regarding the project, please ask them in the discussion session.