ENGG1340/COMP2123 Programming technologies and tools

Mini Project – C++ threads and race conditions

In this mini project, you will be guided to self-learn some new concepts and write a small program in C++ that uses these concepts (threads). Besides our primary instructions, you are encouraged to do self-learning on these new concepts and seek for further tutorials on how to use them. **Note that contents in this self-learning mini project will not show in the final exam.**

Problem Description

You are required to implement a Game of Cell-Culture program that operates in the following way: the game takes a list of gene seeds (Integer N) as input, and a bunch of genesis cell threads are produced according to the input. The genesis cell will live a certain life time (0.1 + N % 8 seconds) and during its life time and it will bring a certain number ((N – N % 8) / 8) of child cells in the middle (1/2) of its life time. Each child cell should live the same life time as its parent cell, but no child is brought. After life time expires, a cell will be dead.

A cell-culture monitor is started just before the first genesis cell is created. The monitor will print out the number of existing live cells every second, so you can continuously monitors how many cells are live.

Program Instructions

1. A main function waits input from user. When the first inputs are given, it will start the monitor thread and then start the genesis cells threads.

2. The number of live cells is counted by a global counter which is shared by all the threads. Note that the counter should be accessed with mutex (see later sections), which means at any time, only one thread can modify this counter.

3. The monitor thread will read the counter every second, and print the number of current live cells.

4. You can use C++ `std::this_thread::sleep_for` to simulate a cell's life time.

Inputs and Outputs Format

```
./cellculture

[Main] Please input a list of gene seeds:

1 2 3 4

[Monitor] Started [ 0 s ]

[Monitor] Total cells: 4 [ 1 s ]

...
```

C++ Threads

Here are some online resources that tells you how to use C++ threads. You would like to read Part 1and Part 2 to learn about how to use threads. Part 3 is related to how to pass initialization values to a thread. Part 4 and Part 5 tells you how to use a global counter while avoiding race conditions. You will learn what is race condition, an important concept in modern computer systems. Moreover, there are consequent tutorials that teach you how to use event handling. You can learn subsequent tutorials by yourself (Part 6-9).

C++11 Multi-threading Part 1: Three Ways to Create Threads

C++11 Multi-threading Part 2: Joining and Detaching Threads

C++11 Multi-threading Part 3: Passing Arguments to Threads

C++11 Multi-threading Part 4: Sharing Data & Race Conditions

C++11 Multi-threading Part 5: Fixing Race Conditions using mutex

Submission and sample test cases

The sample program and test cases will be released after Nov 10, 2019, via moodle. Your submission should be in the form of UID.zip (e.g., 30312342222.zip). Note that it should exactly be a ZIP file. The ZIP file directly (no sub-level directory) contains your source codes and a Makefile, which makes your program to an executable. The filename of the final executable should be cellculture. Incorrect submission formats might result in a zero mark.