Sabancı University

Faculty of Engineering and Natural Sciences

CS204 Advanced Programming

Fall 2019-2020

Homework 8

Multithreading

Due: **12/12/2019, 11:55pm**

No late submissions.

PLEASE NOTE:

Your program should be a robust one such that you have to consider all relevant programmer mistakes and extreme cases; you are expected to take actions accordingly!

You HAVE TO write down the code on your own.

You CANNOT HELP any friend while coding.

Plagiarism will not be tolerated!

Introduction

In this homework, you are going to practice multithreading by simulating a Relay Race. A relay race is a track and field event in which athletes run a pre-set distance carrying a baton before passing it onto the next runner. In the homework, first, runners of the teams will wait for a start signal to start the race together. Then the threads will start and when a runner in the team reaches to destination, the next runner will start. After the last runner in the team reaches to destination, the rank of the team will be updated. Finally, when all teams finish the race, the scoreboard will be printed.

In the homework, you have a struct for **Runners** as follows:

```
struct Runner {
public:
	// VARIABLES  // DEFINITIONS
	float speed;  // Runner's speed
	int rid;      // Runner's ID
	int tid;      // Runner's team ID
	bool reached; // Whether runner has reached destination
	thread* t;    // Runner's thread handle

	// Constructor
	Runner(int rid, int tid):speed(random(4, 8)), rid(rid),tid(tid),reached(false){
		//TODO: Start the corresponding thread here in the constructor

		// Printing information about it
		print.lock();
		cout << "*******\nTeam id: " << tid << "\tRunner id: " << rid << "\nSpeed:
" << speed << "\n*******\n";
		print.unlock();
	}
```

```
};
```
There is also another struct which is for **Teams:**

```
struct Team {
public:
	vector<Runner> teamList;
	bool finished;
	int rank;
	int count;
	int currentRacer;

	Team(int count):count(count), currentRacer(0), finished(false) {}
};
```

You are also given **main** function, **start** function that threads will run and a helper function, **random,** for assigning random speed values to runners. You don't need to make any change in the main function, however you are free to edit any part of the given code as long as it works as it should be.

The parts you need to implement is commented with **TODO** notes. The parts you are responsible for:

- Start the corresponding thread in the constructor of **Runner**
- Update the runner's travelled distance in the **start** function
- Implement the mechanism to start the next thread (next runner in the team) in the **start** function
- implement a function to print the rankings

Output

The output for the given main function should be as follows (**Don't forget the ranking may be different since we are working on threads, don't expect them to be exactly the same**):

```
*******
Team id: 0      Runner id: 0
Speed: 4.005
*******
*******
Team id: 1      Runner id: 0
Speed: 6.25434
*******
*******
Team id: 2      Runner id: 0
Speed: 4.77322
*******
*******
Team id: 3      Runner id: 0
Speed: 7.23496
*******

**************************************************************************
*
Race starting with the signal now
**************************************************************************
*
*******
```

```
Team id: 3      Runner id: 1
Speed: 5.65618
*******
*******
Team id: 1      Runner id: 1
Speed: 7.78698
*******
*******
Team id: 2      Runner id: 1
Speed: 6.78121
*******
*******
Team id: 0      Runner id: 1
Speed: 6.70345
*******
*******
Team id: 1      Runner id: 2
Speed: 5.71648
*******
*******
Team id: 3      Runner id: 2
Speed: 5.98675
*******
*******
Team id: 2      Runner id: 2
Speed: 6.71468
*******
*******
Team id: 0      Runner id: 2
Speed: 4.25098
*******
Team 1 has finished the race
Team 3 has finished the race
Team 2 has finished the race
Team 0 has finished the race


---- RANKS ----

[1] - Team 1
[2] - Team 3
[3] - Team 2
[4] - Team 0
Press any key to continue . . .
```

Some Important Rules

In order to get a full credit, your programs must be efficient and well presented, presence of any redundant computation or bad indentation, or missing, irrelevant comments are going to decrease your grades. You also have to use understandable identifier names, informative introduction and prompts. Modularity is also important; you have to use functions wherever needed and appropriate.

When we grade your homeworks we pay attention to these issues. Moreover, in order to observe the real performance of your codes, we are going to run your programs in *Release* mode and **we may test your programs with very large test cases**.

What and where to submit (PLEASE READ, IMPORTANT)

You should prepare (or at least test) your program using MS Visual Studio 2012 C++. We will use the standard C++ compiler and libraries of the abovementioned platform while testing your homework. It'd be a good idea to write your name and last name in the program (as a comment line of course).

Submissions guidelines are below. Some parts of the grading process are automatic. Students are expected to strictly follow these guidelines in order to have a smooth grading process. If you do not follow these guidelines, depending on the severity of the problem created during the grading process, 5 or more penalty points are to be deducted from the grade.
Name your cpp file that contains your program as follows:

"SUCourseUserName_YourLastname_YourName_HWnumber.cpp"

Your SUCourse user name is actually your SUNet username that is used for checking sabanciuniv e-mails. Do NOT use any spaces, non-ASCII and Turkish characters in the file name. For example, if your SUCourse user name is valent, name is Valentina, and last name is Tereşkova, then the file name must be:

Valent_Tereskova_Valentina_hw1.cpp

Do not add any other character or phrase to the file name. Make sure that this file is the latest version of your homework program. Compress this cpp file using WINZIP or WINRAR programs. Please use "zip" compression. "rar" or another compression mechanism is NOT allowed. Our homework processing system works only with zip files. Therefore, make sure that the resulting compressed file has a zip extension. Check that your compressed file opens up correctly and it contains your cpp file.

You will receive no credits if your compressed zip file does not expand or it does not contain the correct file. The naming convention of the zip file is the same as the cpp file (except the extension of the file of course). The name of the zip file should be as follows:

SUCourseUserName_YourLastname_YourName_HWnumber.zip

For example zubzipler_Zipleroglu_Zubeyir_hw1.zip is a valid name, but

hw1_hoz_HasanOz.zip, HasanOzHoz.zip

are **NOT** valid names.

Submit via SUCourse ONLY! You will receive no credits if you submit by other means (e-mail, paper, etc.).

Successful submission is one of the requirements of the homework. If, for some reason, you cannot successfully submit your homework and we cannot grade it, your grade will be 0.

Good Luck!
CS204 Team (M. Yusa Erguven, Kamer Kaya)