CSCI-440 Assignment 3

Due Midnight on Thursday, March 30

Overview

You are a gladiator. Well … actually you are tasked with writing a gladiator program. In class on Friday, March 31, your gladiator program will compete against your classmates' gladiator programs in a bracketed competition to determine the greatest gladiator of all.

Your Submission

You gladiator program must reside in a single c source code file. You must use your BC username as the name of the file. For example, rmcgregor.c would be mine. The program must be compilable using the following shell command, where username is replaced with your BC username and executable_name is a name of your choosing.

```
$gcc username.c -o executable_name
```

I reiterate, **all of your source code must reside in a single source code file** that has the same name as your BC username.

When your program is compiled for the competition it will be compiled using the command above. You must also provide a file named README that has a single word in it – the name you want to be given to your executable when it is compiled. It must be a valid executable name. This name will be displayed on the screen when your gladiator competes.

Your source code file must contain the following:

- Your name and the date at the beginning of the file
- Proper style, including proper variable names and indentation
- Proper and adequate comments

Please upload to GitHub your single source code file and your README file in a folder named **hw3**.

Your program must accept 3 integer arguments in the order that follows:

1. an integer specifying a semaphore id,
2. an integer specifying your player's id, and
3. an integer specifying the length of one of the side of the game field.

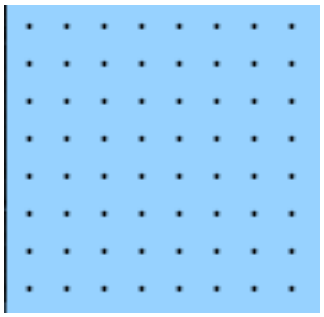
The Arena

Your gladiator will be run against another student's gladiator in a program that I have written which I've called the **colosseum**.

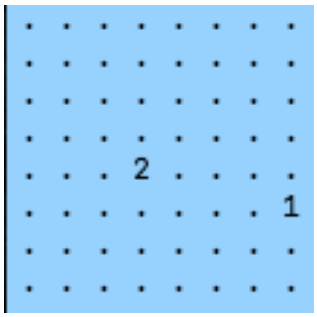
The colosseum is responsible for setting up the game field and forking off two child process in which your gladiator and your opponent's gladiator will be run.

Before starting the competitors, the colosseum allocates an array of semaphores whose length is a perfect square (4, 9, 25, 36, …). Since the length is a perfect square, the array can be viewed as a 2D array. Each of the semaphores is initialized to 0.

The figure below illustrates an array of semaphores viewed as a 2D array (with side length 8), where each dot represents a semaphore with a value of 0.



One of the gladiators, at random, will be assigned the player id 1 and a random semaphore will be set to the value 1 indicating that the gladiator currently *owns* that location. The other gladiator will be assigned the player id 2 and a different random semaphore will be set to the value 2 indicating that gladiator 2 currently owns the location. This is illustrated in the figure below.



Once the gladiators are given one semaphore each, the colosseum will run each of the gladiators in its own process.

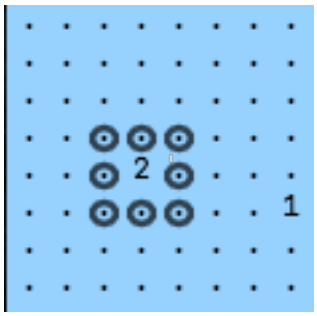
Winning the Game

As mentioned above, each gladiator executable will be passed 3 integers: a semaphore id, a player id, and the side length of the game field.

Since each gladiator has the semaphore id (semid) for the game field and also knows the size of the game field, it can manipulate the game field by changing the values of the semaphores. **The goal** for each gladiator is to cover the game field with its player id – but there are rules.

A gladiator *owns* a semaphore if the semaphore's value is equal to the player's id. A gladiator can change the value in a semaphore to its player id, even if it is currently owned by its opponent, so long as the semaphore that is to be changed is adjacent to a semaphore that they *own* prior to changing the value.

For example, in the figure below, gladiator 2 can change the value of all of the semaphores that are circled.



If a gladiator realizes that they no longer own any semaphores they **must** submit by printing message to standard out indicating that they submit and then terminate.

The contest will continue until a gladiator submits or 2 minutes has expired, whichever comes first. If time expires with no victor, the contest will be considered a draw.

Testing Your Gladiators

I have provided you a program named colosseum.c for you to test your gladiator on. A similar program will be used during the competition. In addition, I have provided a skeleton gladiator program, named gladiator.c, that you may use as a starting point for writing your gladiator program.

These programs are in a compressed archive file named **hw3.tgz**. You should have received this file along with this document. FTP the archive file over to cs.bridgewater.edu, then move the file into your GitHub repository directory.

Once the file is in your GitHub repository directory, uncompress the file using the following command.

```
$tar -xzvf hw3.tgz
```

The tar command will uncompress the archive and create a directory named hw3. To view the files, change your working directory to hw3.

Along with the colosseum and gladiator source code is a make file. The make file will compile both programs and name the colosseum executable **colosseum** and name the gladiator executable **gladiator**.

To compile the both programs, run make on the command line.

```
$make
```

To run the colosseum program with the skeleton gladiator program, type the command given below on the command line.

```
$./colosseum gladiator gladiator
```

After setting up the arena and starting the gladiator programs, the colosseum will display the arena every 0.5 seconds.

To terminate the colosseum program, press **CTRL + c**.

Don't Forget

To reiterate a second time, prior to submitting you code, you must name your gladiator source code file to reflect your username and your program must be able to compile using only the command below.

```
$gcc username.c -o executable_name
```

Closing Comments

The goal of this assignment is to learn how to check the value of semaphore and modify them – not to win the contest. It is hoped, however, that the gaming environment will inspire you to work hard, be creative, and learn how to use semaphores.