Introduction

In this homework, you are asked to implement a safe queue structure and associated functions to be used by threads. The threads will be used to find divisors of some random positive numbers. You will also use command-line options to change parameters.

Queue Structure

Queue is an abstract data type in which the elements are inserted to the rear and are removed from the front (first in, first out). In this homework, it will be implemented as a dynamically allocated array size of which is selected from the command-line. The array will be used circularly. The actual C struct which holds the queue should include these fields:

- The `int` pointer holding the memory location of the array
- The maximum size of the queue
- The current size of the queue
- The index of the front

Any additional variables directly related to the queue operation should also be a field of this struct. If the queue is full, the threads wanting to insert should wait until it is possible. Similarly, if the queue is empty, the threads wanting to remove should wait.

For the operation of the queue, four functions should be implemented, namely:

- `QueueInitialize`: Initialize the necessary fields of the queue.
- `QueueInsert`: Insert an integer to the queue.
- `QueueRemove`: Remove an integer from the queue.
- `QueueDestroy`: Destroy the necessary fields of the queue.

The declarations and implementations of the struct and the related functions should be in `queue.h` and `queue.c` files.

Generator Thread

The program will create one generator thread. This thread will generate random positive integers and insert them to the queue. The quantity of the numbers and their possible range will be decided by the command-line arguments.

Worker Threads

These threads will remove a number from the queue. They will sleep for a random amount of time (0.1 to 1 second with increments of 0.1 seconds). Finally, they will find the divisors of the numbers they removed and print the result. Then, they will start over until the generator finishes working. A result should be as follows:

```
Thread ID: 3059108672, Number: 96, Divisors: 1 2 3 4 6 8 12 16 24 32 48 96
```

Command-line arguments

The program should use four optional arguments to change parameters:

- `-t`: Number of worker threads (default 5)
- `-q`: The maximum size of the queue, which is shared by the threads (default 10)
- `-r`: The amount of the random numbers (default 20)
- `-m`: The possible range of the random numbers (default 100)

Instead of parsing the command-line arguments directly, you may want to use `getopt()`.

Remarks

- There should be **no deadlock or starvation.**
- Synchronization variables should be used **only for critical sections.**
- There should be **no memory leak.**
- Global variables should not be used.
- You can compile your code with this command:

  ```
  gcc -o divisor main.c queue.c –pthread
  ```

- You can find information about headers, functions and types here.
- Send only the source code (`queue.h, queue.c, main.c`). Do not send any executable, since your code will be recompiled.

Examples

```
ee442@ee442-VirtualBox:~/Desktop/HW2$ ./divisor
Thread ID: 3042589504, Number: 29, Divisors: 1 29
Thread ID: 3067767616, Number: 24, Divisors: 1 2 3 4 6 8 12 24
Thread ID: 3050982208, Number: 44, Divisors: 1 2 4 11 22 44
Thread ID: 3076160320, Number: 80, Divisors: 1 2 4 5 8 10 16 20 40 80
Thread ID: 3059374912, Number: 73, Divisors: 1 73
Thread ID: 3059374912, Number: 2, Divisors: 1 2
Thread ID: 3067767616, Number: 56, Divisors: 1 2 4 7 8 14 28 56
Thread ID: 3067767616, Number: 43, Divisors: 1 43
Thread ID: 3042589504, Number: 71, Divisors: 1 71
Thread ID: 3067767616, Number: 55, Divisors: 1 5 11 55
Thread ID: 3076160320, Number: 70, Divisors: 1 2 5 7 10 14 35 70
Thread ID: 3050982208, Number: 40, Divisors: 1 2 4 5 8 10 20 40
Thread ID: 3059374912, Number: 42, Divisors: 1 2 3 6 7 14 21 42
Thread ID: 3050982208, Number: 61, Divisors: 1 61
Thread ID: 3042589504, Number: 80, Divisors: 1 2 4 5 8 10 16 20 40 80
Thread ID: 3076160320, Number: 99, Divisors: 1 3 9 11 33 99
Thread ID: 3067767616, Number: 56, Divisors: 1 2 4 7 8 14 28 56
Thread ID: 3050982208, Number: 77, Divisors: 1 7 11 77
Thread ID: 3059374912, Number: 27, Divisors: 1 3 9 27
Thread ID: 3042589504, Number: 92, Divisors: 1 2 4 23 46 92
```

```
ee442@ee442-VirtualBox:~/Desktop/HW2$ ./divisor -t 3 -q 5 -r 10 -m 1000
Thread ID: 3059592000, Number: 479, Divisors: 1 479
Thread ID: 3067984704, Number: 665, Divisors: 1 5 7 19 35 95 133 665
Thread ID: 3076377408, Number: 154, Divisors: 1 2 7 11 14 22 77 154
Thread ID: 3059592000, Number: 269, Divisors: 1 269
Thread ID: 3067984704, Number: 501, Divisors: 1 3 167 501
Thread ID: 3067984704, Number: 591, Divisors: 1 3 197 591
Thread ID: 3076377408, Number: 998, Divisors: 1 2 499 998
Thread ID: 3076377408, Number: 683, Divisors: 1 683
Thread ID: 3059592000, Number: 254, Divisors: 1 2 127 254
Thread ID: 3067984704, Number: 869, Divisors: 1 11 79 869
```

```
ee442@ee442-VirtualBox:~/Desktop/HW2$ ./divisor -r 5
Thread ID: 3075828544, Number: 40, Divisors: 1 2 4 5 8 10 20 40
Thread ID: 3067435840, Number: 56, Divisors: 1 2 4 7 8 14 28 56
Thread ID: 3042257728, Number: 29, Divisors: 1 29
Thread ID: 3050650432, Number: 73, Divisors: 1 73
Thread ID: 3059043136, Number: 24, Divisors: 1 2 3 4 6 8 12 24
```