Assignment 3: The Barbershop

Due Date: Turn this in before Friday, November 4th at 11:59 p.m.

For this assignment, you will make an implementation of the famous "sleeping barber" problem.

Be careful: there are many descriptions of the sleeping barber problem online. You are free to read about them, but there are many subtle variations in the way the problem works.

The Sleeping Barber Problem

- A barbershop has three waiting chairs, and the barber chair.
- If there are no customers, the barber sits in the barber chair and sleeps.
- If a customer enters the shop and no waiting chairs are available, the customer leaves.
- If the barber is busy, but chairs are available, the customer sits in a chair.
- If the barber is asleep, the customer wakes up the barber.

The Code I've Given You

The code I've given you creates a barber thread (running the `barber_thread`) function. It creates an additional customer thread (running the `customer_thread` function) every time you press a key.

You shouldn't need to write any grahpics code or alter the `main` function in any way. I've provided simple functions that allow each thread to manipulate a single shape on screen. For example, when a thread calls:

```
capp::enter();
```

That creates a shape for the thread and moves it onto the screen. If that thread then calls:

```
capp::move_to({x, y});
```

then its shape will move to row `x`, column `y`.

What You Do

You need to add code so that the barber and customer threads act according to the description above.

- No thread should ever busy-wait! For example, customer threads sitting in the waiting chairs should probably be waiting on a condition variable.
- Whenever a thread is waiting, its shape should be red.
- Whenever a thread is not waiting, its shape should be green.
- When the barber sleeps, he goes to his chair (`BARBER_CHAIR`).
- The barber should call `cut_hair()` when a customer is in the barber chair.
- The customer shouldn't leave the chair until the haircut is complete.
- The barber thread should run in a loop. Neither it nor the customer threads need to check for program termination.
- You can't use semaphores for this assignment.

Turn it in

- You should modify the source code I've provided you. There is no need for extra source files or changes to the Makefile.
- Make a folder labeled `Assignment 3` in your turn-in directory and copy in your source files and Makefile.