Locking Workshop

Loop

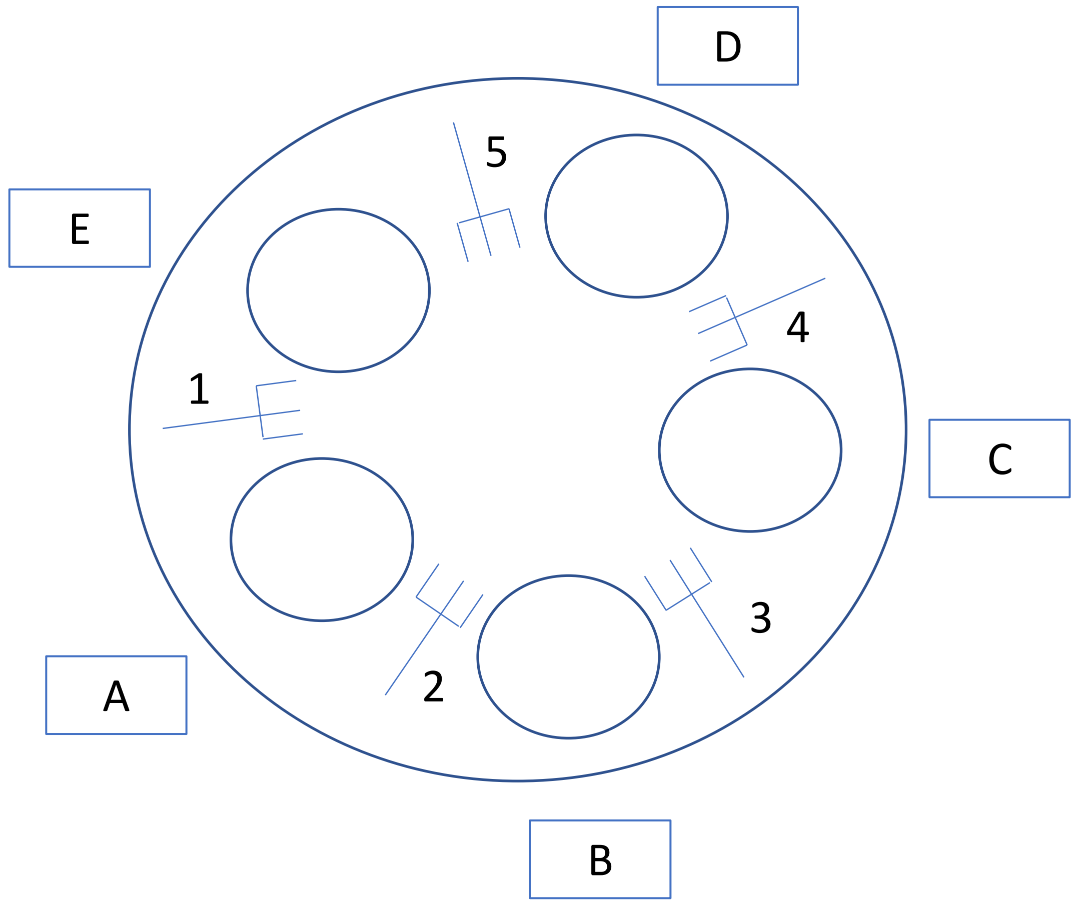
- Imagine the following function is executed concurrently by two threads

```
int x{0};

void func() {
    while (x == 0) {
        x = 1 - x;
    }
}
```

- Are there any possible scenarios in which neither thread is able to exit the loop? If so, how would you fix the problem?

Dining Philosophers



Dining Philosophers problem

- Five philosophers are seated around a table. Each philosopher has a bowl of spaghetti in front of them and a fork at their left hand side
  - A philosopher has two states: thinking and eating
  - A philosopher can only eat when they have both a left and right fork
  - Each fork can only be held by one philosopher at a time

Dining Philosophers problem contd

- A philosopher can only pick up one fork at a time
  - A philosopher may pick up a fork as soon as it is put down by another
  - When a philosopher finishes eating, they must put down both forks immediately
  - A philosopher has no awareness of what the other philosophers are doing
  - If a philosopher does not eat at all, they will die of starvation

Dining Philosophers implementation

- Identify some potential issues in writing a program which runs each philosopher in their own thread
  - No philosophers must be harmed during the execution of this program!
- Describe possible solutions to these issues
- Consider a solution which uses a mutex
  - A philosopher must obtain a lock on this mutex before picking up any forks
  - Are there any drawbacks to this solution?
- Is there any way to implement this without explicitly or implicitly synchronizing the philosophers?