# CS 153
# Design of Operating Systems

## Winter 2016

Midterm Review

# Homework 1

- Homework 1 solution to be posted in ilearn today

# Midterm

- in class on Monday

- Based upon lecture material and Chapters 1 to 7
  - Closed book.  No additional sheets of notes

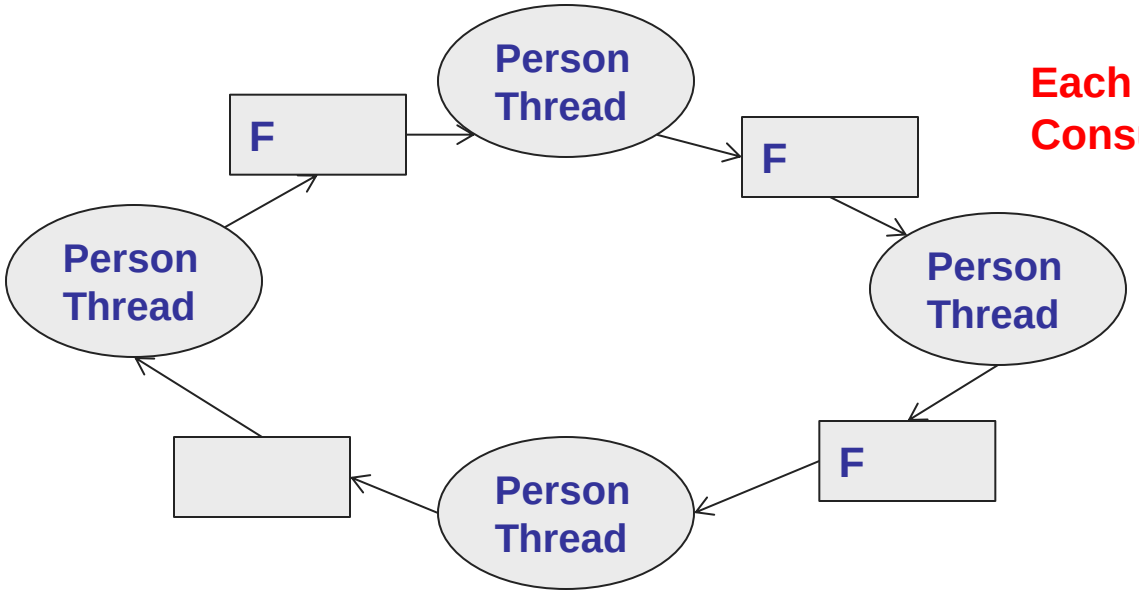
# Lets do some problems

*Problem 4:* (20 points; 12 minutes)
For the 2012 presidential election, you are writing code for real-time processing of the votes from the voting machines across New York State. You use shared counters (one for each candidate) to keep track of the votes as they come from multiple voting machines. You can think of each machine as a thread–every time it receives a vote, it increments a counter for that candidate.
(a) (5 points) What happens if you let the threads run without synchronization?
(b) (8 points) Suggest two ways to use locks to solve the problems you discuss in part (a); which one is more conservative?
(c) (7 points) A CS major that took CS350 suggests the following improvement to the implementation. She suggests that each thread should maintain a local count of the votes, and then update the global count periodically. Do we still need synchronization and where? Why does this implementation outperform the one with only the global count variables?

*Problem 3:* (20 pts) Consider a multiple feedback scheduler with three levels. The first level has a quantum of 5ms, the second has a quantum of 10ms, and the third is FCFS. You have a set of processes with a run time of 9ms, 16ms, 4ms, 20ms, and 7ms that arrive at times 0, 3, 6, 8 and 10 ms respectively. What is the average normalized turnaround time for the processes? Show your work, including the state of the queues whenever they change.

*Problem 5:* (30 pts; 25 minutes) Two or more people are playing a game of frisbee with two or more frisbees. Initially, the frisbees are given to different people and the number of people and frisbees is known. The players stand around in a circle and the frisbees are thrown clockwise along the circle. Each person receives a frisbee and throws it to the next person over. Unfortunately, if a frisbee is thrown at a person when he already has another, he will drop it – we do not want that case to happen.

- (15 points) Implement a pseudocode simulation of this problem (you can use locks, semaphores or condition locks)
- (5 points) Is there a problem if the number of frisbees equals the number of players? Explain




**Each person is both Consumer and Producer**

*Problem 5:* (30 pts; 25 minutes) Two or more people are playing a game of frisbee with two or more frisbees. Initially, the frisbees are given to different people and the number of people and frisbees is known. The players stand around in a circle and the frisbees are thrown clockwise along the circle. Each person receives a frisbee and throws it to the next person over. Unfortunately, if a frisbee is thrown at a person when he already has another, he will drop it – we do not want that case to happen.

- (15 points) Implement a pseudocode simulation of this problem (you can use locks, semaphores or condition locks)
- (5 points) Is there a problem if the number of frisbees equals the number of players? Explain

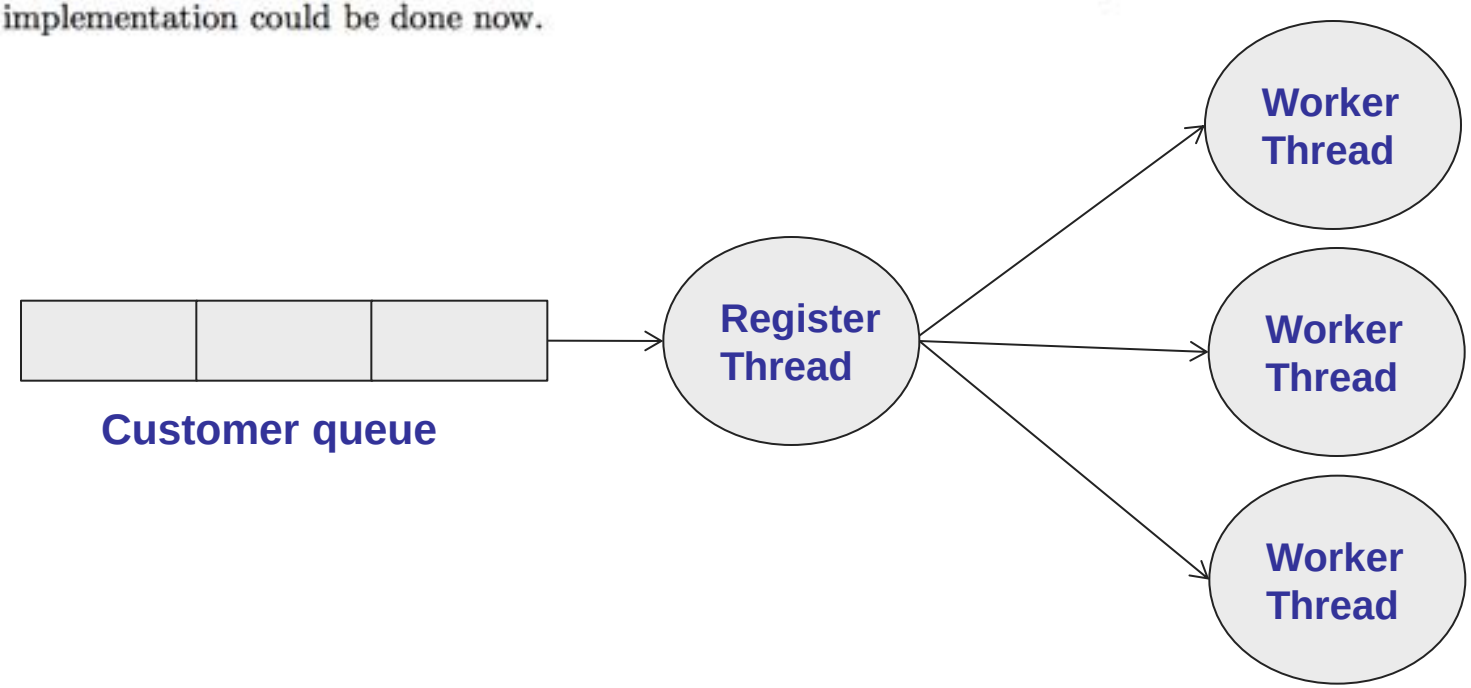
```
Thread() {
   while(true)  {
      // check incoming?
      // receive incoming
      // check outgoing?
      // throw out
   }
}
```

```
Thread_i() {
   while(true)  {
      wait(full_i);
      //receive incoming
      signal(empty_i);
      wait(empty_(i+1));
      // throw out
      signal(empty_(i+1));
   }
}
```

*Problem 6:* (25 pts; 20 minutes) At the coffee shop in the library, people queue up to buy coffee from the register. After placing their order, they go and wait for their coffee. There are three workers, in addition to the person on the cash register, and each works on one order at a time.

(a) Write psuedocode to simulate the coffee shop operation

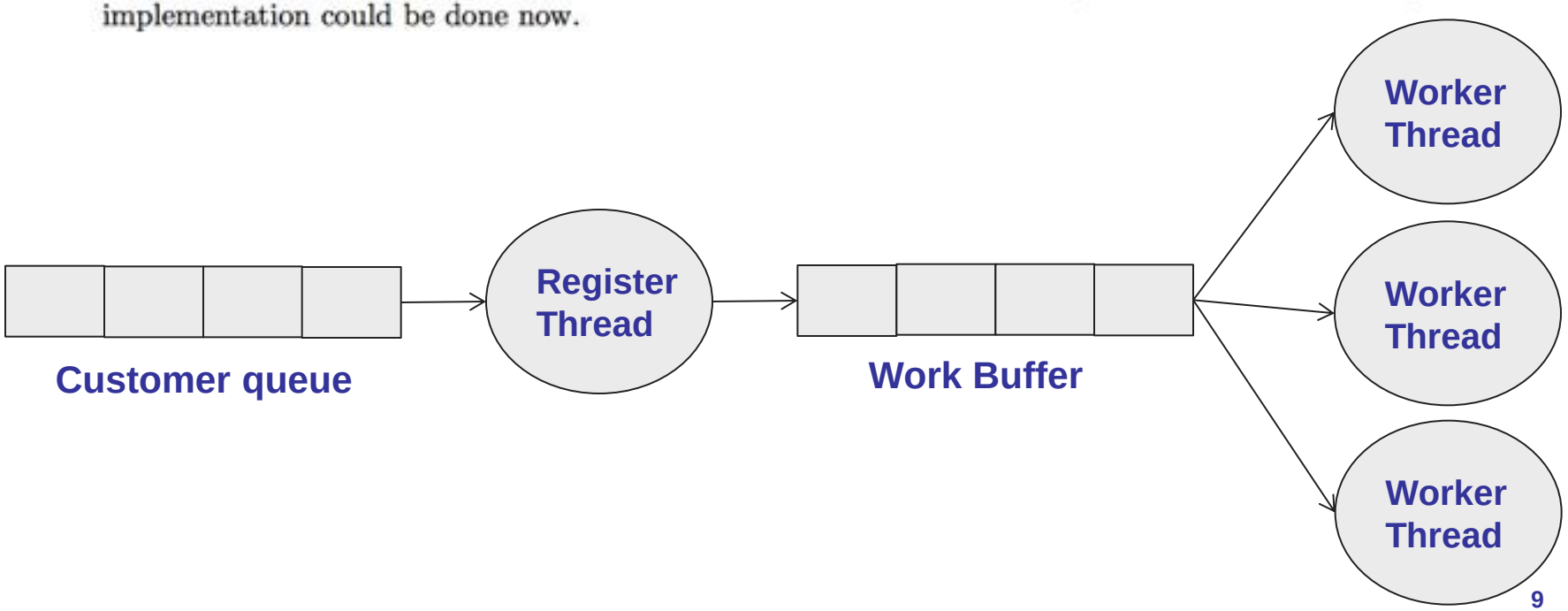
(b) Since there are many types and flavors of coffee, its important that everyone picks up their correct order. Assume now that coffees may not be ready in the order that they were placed – for example, regular coffee is ready much faster than a double shot latte. Explain (in words, but be specific) how the implementation could be done now.



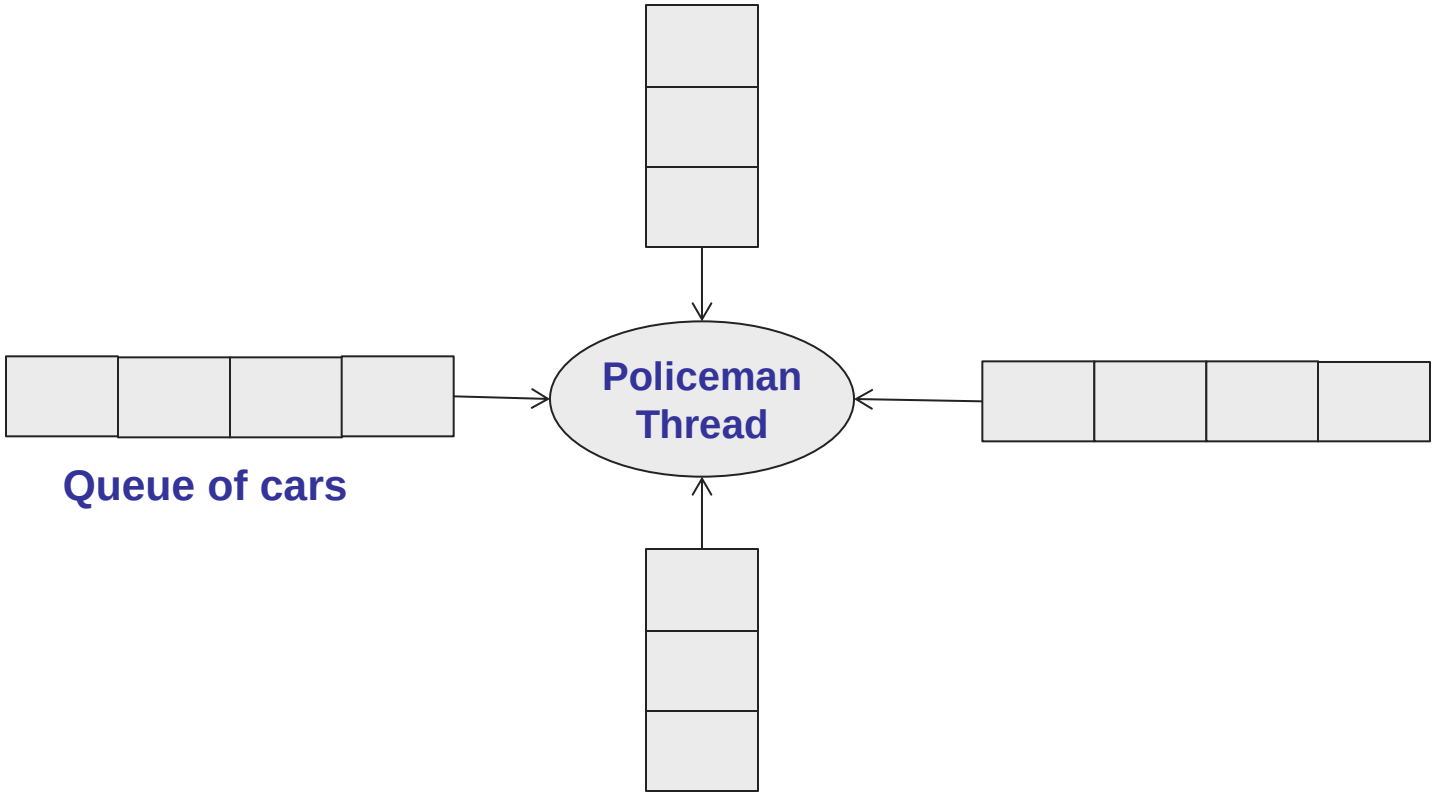
*Problem 6:* (25 pts; 20 minutes) At the coffee shop in the library, people queue up to buy coffee from the register. After placing their order, they go and wait for their coffee. There are three workers, in addition to the person on the cash register, and each works on one order at a time.

(a) Write psuedocode to simulate the coffee shop operation

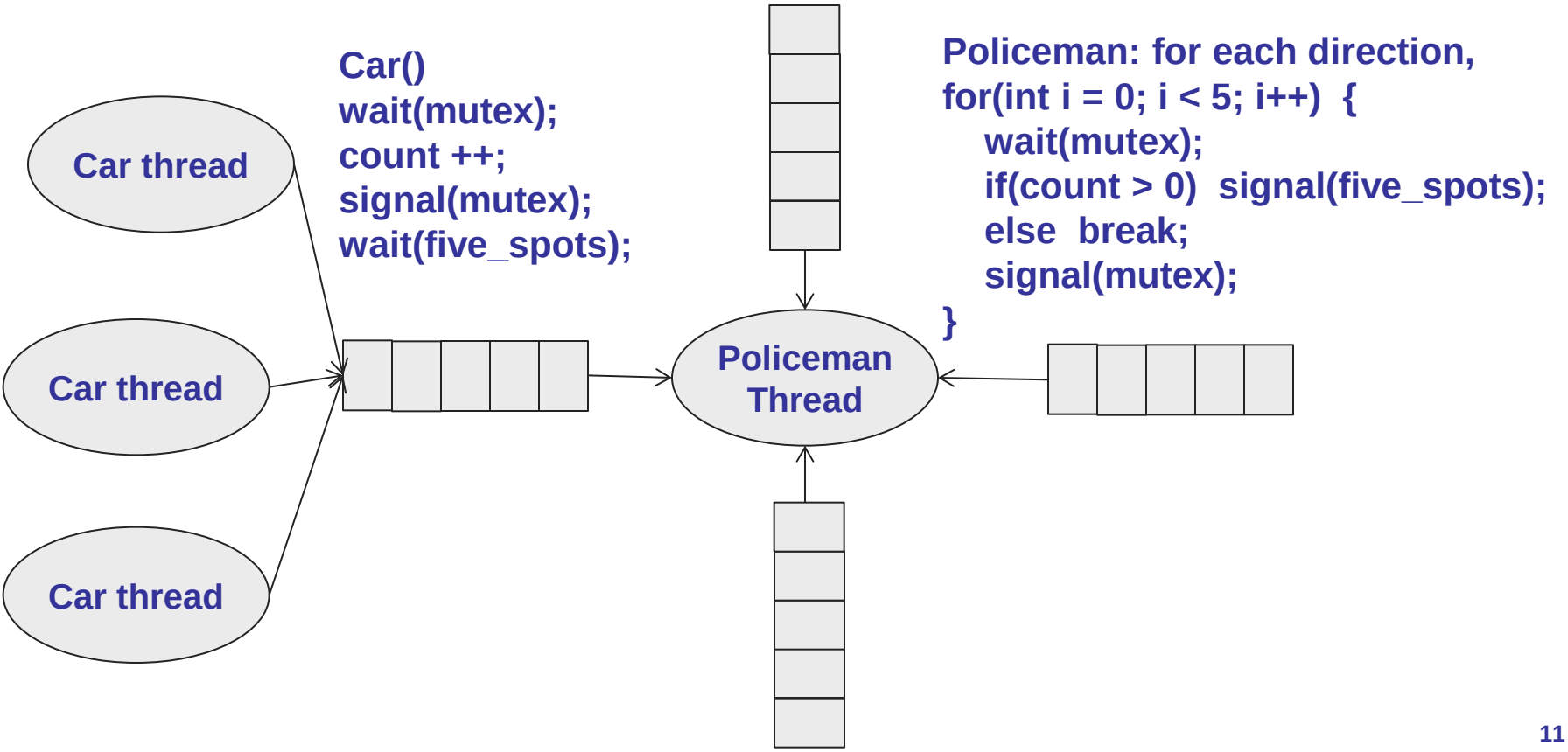
(b) Since there are many types and flavors of coffee, its important that everyone picks up their correct order. Assume now that coffees may not be ready in the order that they were placed – for example, regular coffee is ready much faster than a double shot latte. Explain (in words, but be specific) how the implementation could be done now.



*Problem 5:* (15 pts; 15 minutes) A traffic policeman is in charge of controlling traffic at a busy intersection. Cars arrive from 4 directions (North, East, South and West). The policeman goes through the directions in the order they are listed above (N, E, S, W then back to N). When he starts a new direction, he allows at most 5 cars to go; if there are less than 5 he changes to the next direction after the last car passes through. Show the psuedo-code for the policeman and the cars to simulate this problem.



*Problem 5:* (15 pts; 15 minutes) A traffic policeman is in charge of controlling traffic at a busy intersection. Cars arrive from 4 directions (North, East, South and West). The policeman goes through the directions in the order they are listed above (N, E, S, W then back to N). When he starts a new direction, he allows at most 5 cars to go; if there are less than 5 he changes to the next direction after the last car passes through. Show the psuedo-code for the policeman and the cars to simulate this problem.




```
Car()
wait(mutex);
count ++;
signal(mutex);
wait(five_spots);
```

```
Policeman: for each direction,
for(int i = 0; i < 5; i++)  {
    wait(mutex);
    if(count > 0)  signal(five_spots);
    else  break;
    signal(mutex);
}
```