CS241#20 – CSP II. Race Conditions. Deadlock II and Dining Philosophers

1. Is it necessary for the change() method to lock the mutex, to release a blocked thread?

|  |  |
| --- | --- |
| change() {  x = 1  pthread\_cond\_signal(&cv)  } | void wait\_for\_positive\_x() {  pthread\_mutex\_lock(&m);  while(x < 1)  pthread\_cond\_wait(&cv, &m);  pthread\_mutex\_unlock(&m);  } |

... Implications for cond\_wait implementation? Implications for the firework code?

2. Do we have a winner for the CRITICAL SECTION PROBLEM? Contestant #4:

Three shared variables: turn = 1, flagA = FALSE, flagB = False

|  |  |
| --- | --- |
| thread1:  flagA = TRUE  if(flagB) while(turn==2){ /\* check again \*/}  // Do Critical Section stuff  turn = 2  flagA = FALSE | thread2:  flagB = TRUE  if(flagA) while(turn==1){ /\* check again \*/}  // Do Critical Section stuff  turn = 1  flagB = FALSE |

3. Deadlock

The \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ conditions for deadlock are:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_: "A process is currently holding at least one resource and requesting additional resources which are being held by other processes."

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_:"There is a set of waiting processes, such that P1 is waiting for a resource held by P2, P2 is waiting for a resource held by P3 and so on until PN is waiting for a resource held by P1."

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_:"A resource can be released only voluntarily by the process holding it, after that process has completed its task"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_:"At least one resource must be held in a non-shareable mode"

Three gardeners visit the garden shed pick up their desired tools for the day. There is a potential for deadlock. Fortunately they know about the C\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ conditions! Find four ways to solve the problem (break one condition each time). Name which condition you break in each case.

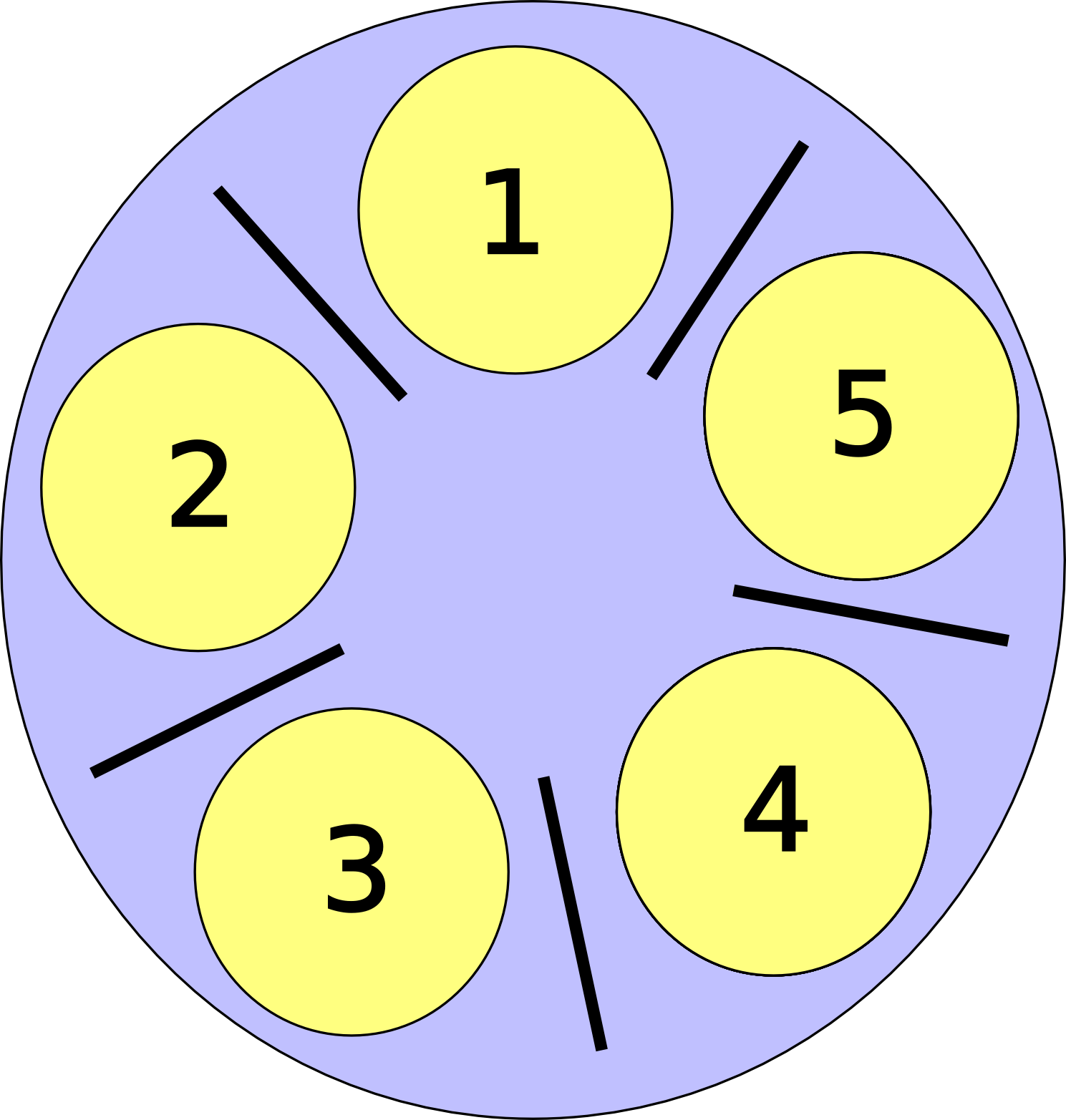
1

2

3

4

Remember Mergesort? How can you implement parallel Mergesort? Explain what synchronization calls you will use and when.

What is the Dining Philosophers problem?

Candidate Solutions:

1. "Pick up left chopstick. Pickup right chopstick. Eat. Release both."

2. "Pick up right. Pick up left. Eat. Release both"

3. "Eat when I tell you"

4. "Pick up left chopstick. Try to pickup right chopstick (Fail? release both and restart). Eat. Release both."

5?