# CS241 Lawrence Angrave L16 – Condition Variables II. Implementing a barrier. Producer Consumer

1. Condition Variable pop-quiz
Which call do you implement inside a loop?
What is spurious wake up?
Why do you need a mutex too?
How do you wake up one or all blocked threads?

2. Fix the following multithread code. remove should never allow the account to go negative.

```
pthread_mutex_t m;
pthread_cond_t cv;

int money = 100;

void init() {
  money = 100;

}

void add(int amount) {

  money += amount;

}

int remove(int amount)


  money -= amount;
  return money;
}
```

3. Three classic / well known synchronization problems:

Barrier

Producer Consumer

Reader-Writer Problem

4. Use a CV to implement a simple version of a *counting semaphore*
Note a real semaphore might implement a queue of waiting threads to ensure fairness (and avoid *starvation*).

| | |
|---|---|
| `sem_init(sem_t *s, int shared, int value) {`<br><br><br><br>`}` | `typedef struct sem_t {`<br><br><br><br>`} sem_t;` |
| `sem_post(sem_t*s) {`<br><br><br><br><br>`}` | `sem_wait(sem_t*s) {`<br><br><br><br><br>`}` |

5. Use a CV to implement a *barrier* Do not continue to calc #2 until all 16 threads have reached the barrier.

```
pthread_mutex_t m;

?_____________________
double data[256][8192] ;
int main() {
    pthread_mutex(&m, NULL);

    ?_____________________
    pthread_t ids[N];
    for(int i=0;i<N;i++)  pthread_create( ?____________ , NULL , calc, (void*) i );
    // Wait for all threads to finish
    for(int i=0;i<N;i++)  ?_____________________

    /* print out result*/
}

?____  calc( ?___________________   ) {
    /* Divide matrix work up into blocks of 16 columns.

   int x,y, start = 16 *  ?_____________________
   int end = start + 16;
   for(x = start; x<end;x++) for(y=0; y <8192;y++) /* do calc #1 */
   // Wait here until all threads have finished calc #1.









    for(x = start; x<end;x++) for(y=0; y <8192;y++) /* do calc #2 */

    return ?______
}
```