CS241 #16 CV Cookies, Counting Semaphores, Ring buffer.

1. Condition Variables Warm-up Challenge: Eat cookies fast!

Meanwhile in a Parallel Universe …

Two threads viciously eat cookies but are blocked on a c.v. …

```
01 int jar = 0;
02 pthread_mutex_t m = PTHREAD_MUTEX_INITIALIZER;
03 pthread_cond_t cv1 = PTHREAD_COND_INITIALIZER;
04
05 void* cookie_eater(void*arg) {
06  char* name = (char*) arg;
07  while(game_running) {
08    while(jar == 0) {
09       printf("%s nap time\n", name);
10       ?

11    }
12    jar --;
13    printf("%s eats! %d remain\n",name, jar);
14  }
15  printf("%s is exiting...",name);
16  return NULL;
17 }
```

Complete the `add_cookies` to add cookies to the cookie jar
(Pretend cookie jar has ∞ capacity)

```
18 void add_cookies(int add) {
19   assert(add > 0);
20
21
22
23
24
```

2. What must be locked before calling `p_cond_wait` ? _____

3. You wake a thread blocked inside a condition variable but it does not return from `p_cond_wait`. Why?

Another thread still _____

The blocked thread will continue when _____

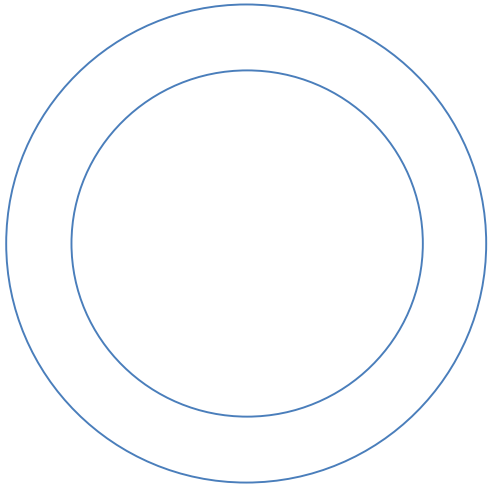
4. How do I use counting semaphores?

`sem_init`

`sem_wait`

`sem_post`

5. What is a fixed ring buffer? Why would I use it?



6. Producer Consumer Case Study:
Use counting semaphores to implement a fixed ring buffer

```
pthread_mutex_t m;

  // (Not OSX!)

sem_t  _________________

void init() {
  sem_init(____________ ,  0, ______);
  sem_init( ____________,  0, ______);
  pthread_mutex_init( &m , NULL);
}

void sync_enqueue(work_t *work) {









}
work_t* sync_dequeue(){
```

7. Quick quiz

i) How many threads can be executing line 8 or 14 at a time? Why?

ii) What have I made? (Add missing code + Suggest function names for A and B)

```
01  pthread_mutex_t m = PTHREAD=MUTEX_INITIALIZER;
02  pthread_cond_t cv1 = PTHREAD_COND_INITIALIZER;
03  int  mystery = 5;
04
05  void  A?() { // Waits if count would become -ve
06    p_m_lock(&m)
07    while(mystery == 0) p_cond_wait(&cv1, &m);
08    mystery --;
09    p_m_unlock(&m);
10  }
11
12  void B?() {
13     p_m_lock(&m);
14     mystery ++;
15     if( _____________ ) p_cond_signal(&cv1);
16     p_m_unlock(&m);
17  }
```