CS214 Recitation

0 How do I block a thread (=send it to 'sleep')?

1. How do I wake up threads that are blocked on a condition var?

2. The cake is a lie... Complete the following methods using a condition variable and mutex locks. The cake integer must never be negative.

```
01   pthread_mutex_t m = PTHREAD=MUTEX_INITIALIZER;
02   pthread_cond_t cv = PTHREAD_COND_INITIALIZER;
03
04   int  cake = 0;
05
06   void  decrement() { // Waits if nonzero
07
08     while(cake == 0) {
09       // sleep

10
11     }
12     cake --;
13
14   }
15
16   void increment() {
17       cake ++;
18   }
19
```

3. How does pthread_cond_wait *really* work?

---

1. Producer Consumer (review)
Assume buffer is an array of length 16.

add(value) {	remove() {
sem_wait(&sem_empty)	sem_wait(&sem_full);
buffer[(in++) & 15] = value;	result = buffer[(out++) & 15];
sem_post(&sem_full);	sem_post(&sem_empty);
Q. What are 'sem_empty' and sem_full? When do they block? Q. What should be their initial values?	

Q. What if sem_full was only initialized to 7? Would the producer consumer still work? 32?

Q. What is missing from the above code? When would it matter?

Q. Could you implement a producer consumer queue using condition variables instead?

Q. Can you make a queue of work items for threads?

2. Fix the following multithread code to thead safe. Additionally, remove should never allow the account to go negative i.e. it should block until there are sufficient funds.

```
pthread_mutex_t m;
pthread_cond_t cv;

int money = 100;

void init() {
   money = 100;

}

void add(int amount) {
  assert(amount>0);

  money += amount;

}

int remove(int amount)

   money -= amount;
   return money;

}
```

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 P_1 is waiting for a resource held by P_2, P_2 is waiting for a resource held by P_3 and so on until P_N is waiting for a resource held by P_1."

_____:"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"

Dining Philosopher's Problem.

How can one break deadlock?