CS241 #14
Working With threads and locks

0.0 Would you expect the following to work on your 64 bit VM?

```
01  int bad = (int) "Hello";
02  puts( (char*) bad);
```

0.1 Which of the following calls will block?

pthread_mutex_init
pthread_mutex_lock
pthread_mutex_unlock
pthread_mutex_destroy

0.2 You call to *pthread_mutex_X* (what is X?) blocks. When will it return i.e. when will it unblock?

0.3 Why might pthread_mutex_X not block?

Where are the critical sections in the following code examples?
Fix any errors you notice.
Modify the code to be thread safe

```
01
02
03  link_t* head;
04
05  void*list_insert(int v) {
06    link_t* link = malloc( sizeof(link_t*));
07    link -> value = v;
08    link -> next = head;
09    head = link;
10  }
11
12  link_t* list_remove() {
13     link_t* result = head;
14     if(result) head = result->next;
15     return result;
16  }
```

```
01
02  size_t capacity = 64;
03  size_t size = 0;
04  char** data = malloc(capacity);
05
06  void push(char*value) {
07    if(size == capacity) {
08      capacity *= 2;
09      realloc(data,capacity);
10    }
11    data[size++] = value;
12  }
13  char* pop() {
14     char* result = data[--size];
15     return result;
16  }
```

3. Notice any mistakes? What do you expect to happen?

```
01  pthread_t tid1,tid2;
02  pthread_mutex_t m;
03
04  int counter;
05  void*myfunc2(void*param) {
06   int i = 0; // stack variable
07   for(; i < 1000000;i++) {
08     pthread_mutex_lock( &m );
09     counter ++;
10    }
11   return NULL;
12  }
13  int main() {
14   pthread_create(&tid1, 0, myfunc2, NULL);
15   pthread_create(&tid2, 0, myfunc2, NULL);
16   pthread_join(tid1,NULL);
17   pthread_join(tid2,NULL);
18   printf("%d\n", counter );
19  }
```

4. Meet your next *Synchronization Primitive*: What is a *Counting Semaphore*?

5. Case study: Parallelize *AngraveCoin* miner for fun and profit!

```
void search(long start, long end) {
  printf("Searching from 0x%lx to 0x%lx\n", start , end);
  for(long i = start; i < end; i++) {
    char message[100];
    sprintf(message,"AngraveCoin:%lx", i);

    unsigned char *hash; // 256 bit result ( = 32 bytes )

    hash = SHA256(message, strlen(message), NULL);

    int iscoin; // first three bytes must be zero
    iscoin =(hash[0]==0)&&(hash[1]==0)&&(hash[2]==0);

    if(iscoin)
        printf("%lx %02x %02x %02x '%s'\n", i, res[0],
res[1], res[2] , message);
  }
  printf("Finished %lx to %lx\n", start, end);
}

// want to speed up search of 2^33 possible coins
long array[] = {0L, 1L <<25, 1L <<27, 1L <<33};
int main() {
  search(array[0], array[1]);
  search(array[1], array[2]);
  search(array[2], array[3]);
  return 0;
}
```