1. Complete this code to print the thread id and an intial starting value. What does this code actually print?

```
void* myfunc(void*ptr) {
   printf("My thread id is %ld and I'm starting at %d\n");
   return NULL;
}

int main() {
   // Each thread gets a different value of i to process
   pthread_t tid;
   for(int i =0; i < 10; i++) {
      pthread_create(&tid, 0, myfunc, &i);
  }
  ...
```

2. What does this code print? Will it always print the same output?

```
int counter2;

void*myfunc2(void*param) {
  int i=0; // stack variable  - so local to each thread.
  for(; i < 1000000;i++)
     counter ++;
  return NULL;
}

int main() {
     pthread_create(&tid1, 0, myfunc2, NULL);
     pthread_create(&tid2, 0, myfunc2, NULL);
     pthread_join(tid1,NULL);
     pthread_join(tid2,NULL);
     printf("%d\n", counter );
}
// Copy-paste gotcha: &tid1 twice in 2nd p_create.
```

3 Use heap memory to pass starting information to each thread. Create two threads. Each thread will do half the work. The first thread will process 0..numitems/2 in the array. The second thread will process the remaining items. Any gotchas?

```
typedef struct work_ {
?
?

} work_t;

int start_threads(int * data, size_t numitems) {
    size_t half = numitems/2;




    pthread_create(&tid1, 0, imagecalc,?);
}
// Gotcha odd number of numitems?
```

11a Why are some functions e.g. `asctime,getenv, strtok, strerror` not thread-safe?

11b How would you 'fix' this function to be "thread-safe"

```
char* to_message(int num) {
   char static result [256];
   if(num < 1000) sprintf(result, "%d : blah blah" , num);
   else strcpy(result, "Unknown");
   return result;
}
```

12. What are condition variables, semaphores, mutexes?

12b Can you call malloc from two threads?

13. Advantages of threads over forking processes?

14. Can you fork a process with multiple threads?

15. Examples of why you might fork processes

16 If there's time.... Intro to pthread_mutex