Christopher Mai
chrmai1
Section 1
hw3p1

A). Exercise 6.17 Page 270

Semaphores can be used to limit the number of connections. All a server has to do is implement counting semaphores. Basically each connection needs to get a handle on the semaphore and issue a wait command signaling that one of the lines is being used. The semaphore can be restricted to allowing a certain number of availble resources, in this case 5. As each connection gets a handle on the semaphore the count decrements until 0. At 0 no more connections can perform a wait operation on the semaphore preventing additional connections. Also, it is necessary for closing connection to issue the signal command incrementing the semaphore count, thus allowing for a connection to take it's place.

B). Exercise 6.29 Page 271

```
monitor file
{
 // p is however number of prcess are available
 // assumes numbering starts a 0
 access[p];

 condition self[p];

 void access(int i){
  access[i] = -1; // -1 trying to get access.
  test(i);
  if(state[i] != 1){
   self[i].wait();
  }
 }

 // tries to resume other processes that may be waiting;
 void close(int i){
  access[i] = 0;

  for(int x = 0; x < p; x++){
   if(access[x] == -1){
    test[x];
   }
  }
 }

 void test(int i){
  int check = 0;
  for(int x = 0; x < p; x++)
```

```
  {
        if(access[x] == 1){
          check += x;
     }
  }
  // n is the allowed number
  // check is the total number of processes currently     //  accessing
  // i is the number of the file trying to get access

  if((check + i)  < n){
    access[i] = 1;
    self[i].signal();
  }
 }

 init_code(){
  for(int i = 0; i < 5; i++){
        access[i] = 0;  //0 = no access. 1 = access
  }
 }
}
```