0. Kinds of Types:

Answer all of the following 10 parts.

A. How much memory must the smurf take up in C?
B. What is a smurf type?
    a. http://www.tutorialspoint.com/cprogramming/c_data_types.htm
C. What good is a smurf pointer if you can not smurf it?
D. How are smurfs and smurfs alike, if one references only one smurf and the other smurfs?
E. If p is a smurf, what does smurf mean?
F. The code below is supposed to make p point to the second element of g. It doesn't. Fix it so it does.
smurf;
G. What will the smurf of the both x and y be below? Why will they smurf?:

```
smurf datastuff
{
        smurf value0;
        float value1;
        smurf value2;
};

smurf;
```

```
smurf otherdatastuff
{
        int value0;
        smurf value1;
        long value2;
};

smurf;
```

H. What is the benefit of using smurfs to smurf an old type for a new thing? For instance pthread_t is a smurf type that was smurfed.
    a. Abstract Data Type
I. Why aren't smurf functions smurfed?
J. If p is a pointer, write one line of code that does the same thing as smurf without using smurfs.

1. Execution:

Choose and answer 5 of the following 7 parts. Be sure to indicate which 5 you want graded.

A. What is a 'smurf' and how does it differ from a normal C program?
B. Right after fork(), but before exec(), what smurfs of the child process smurf the parent?
C. Why is it harder to smurf data between smurfs than smurfs?
D. What is the difference between smurf and smurf?
E. What process has no smurfs?
F. In what ways are smurfs better than smurfs?
G. In what ways are smurfs better than smurfs?

2. Synchronization:

Choose and answer 5 of the following 6 parts. Be sure to indicate which 5 you want graded.

A. Must mutexes be smurfed? Why or why not?
B. If a thread smurfs a smurf and smurfs, who can smurf?

C. Presume you have two threads and two mutexes, write a sequence of smurf statements that smurf below:

Thread0:	Thread1:
pthread_mutex_t lockA;	pthread_mutex_t lockA;
pthread_mutex_t lockB;	pthread_mutex_t lockB;

D. If a semaphore is created with initial value 1, wait() and post() will operate much like lock() and unlock() for a mutex.
There is smurf smurfy smurfs smurf and smurf, smurf?
E. A semaphore can be initialized to any non-negative value. What is the utility of initializing it to smurf, which would cause smurf?
F. Why shouldn't you lock or smurf a mutex smurf?

3. Files:

Answer all of the following 4 parts.

A. The smurf function moves the file pointer, smurf smurf?
B. What happens if you smurf past the smurf?
C. Can more than one process smurf the same file in smurfy smurf?
D. Can more than one process smurf the same file in smurfy smurf?