CIS 3207. Quiz 3a.

Name:

The quiz is out of 20 points. There are 2 pages.

You're not expected to remember all of the details of the entire API for synchronization. In any one of your solutions, you're welcome to use generic names, *e.g.*, `P(&sem)`, `V(&sem)`, `init(&sem, init_val)`, `wait(&cond, &mutex)`, `signal(&cond)`. If the primitive needs to be initialized, do not forget to show its initial value.

1. [2 points] If I know a hosts name, *e.g.*, Amazon.com, what service (protocol) can provide me with a numeric address?

---

2. [2 points] Concurrency implies parallelism and parallelism implies concurrency.
   A. True B. False

3. [2 points] Suppose that I have a descriptor `fd`, which is attached to a certain file. I then run the code `dup2(fd, STDOUT_FILENO)`, which executes successfully. Circle all that apply.
   A. subsequent writes to the file go only the screen and not the file
   B. subsequent `printf()`s write to the file instead of `stdout`
   C. `STDOUT_FILENO` refers to the entry for the file in the open file table
   D. subsequent writes to the file go to both the file and to the screen

4. [2 points] Name two important disadvantages to user-level threads.

---

5. [2 points] What two pieces of information does a process on one host need to know if it wants to send a message to a process running on another host?

---

6. [4 points] Spongebob just bought a brand new 50" TV. Patrick is going to help him bring it into his pineapple ... house, rather. Patrick shouldn't open the door until Spongebob arrives at home. He should then open the door, and after Spongebob has entered the house, Patrick should shut the door. Similarly, Spongebob should not attempt to enter until the door is open.

   Use synchronization primitives in order to guarantee the proper ordering of events.

Patrick init(door, 0) init(car, 0) **Spongebob**

```
1  P(&car)
2
3  open(&door);
4  V(&door)
5
6
7  close(&door);
8
9
```

```
1
2
3  drive(&car, home);
4   V(&car)
5    P(&door)
6
7  enter(home);
8
9
```

7. [3 points] Is there a race condition in the following code? If so, modify it to eliminate the race.

```
#define N 10000

int i;

int main(void) {

  fork();
  for (i=0; i<N; i++) {

    printf("%d\n", i);

  }

  return EXIT_SUCCESS;
}
```

8. [3 points] Is there a race condition in the following code? If so, modify it to eliminate the race.

```
int x;

void main(void) {
  int i;

  pthread_t threads[N_THREADS];

  x=0;

  for (i=0; i<N_THREADS; i++) {

    pthread_create(&threads[i], NULL, func, NULL);

  }

  for (i=0; i<N_THREADS; i++) {

    pthread_join(threads[i], NULL);

  }
}
void *func(void *arg) {
  int i;

  for (i=0; i<100; i++) {

    x++;

  }

  return NULL;
}
```