Apellidos: **Nombre:**

Concurrencia y Paralelismo

Ingeniería Informática de Sistemas

Examen Junio 2012

1. **Comunicar threads a traves de dos canales de comunicación** [2.5 puntos]

   Partimos de dos colas de elementos: `queue1` y `queue2`. Tenemos dos tipos de threads: `thread1` y `thread2`. Los threads `hread1`, quitan elementos de la `queue1`, le aplican la función `f1` y meten el resultado en `queue2`. La función `thread2` hace el trabajo inverso, quita los elementos de `queue2`, les aplica la función `f2` y mete los resultados en `queue1`. Se da el código de ejemplo y los prototipos de las funciones que se pueden utilizar. El código de ejemplo no realiza ninguna protección de ninguna de las colas. Proteger las dos colas teniendo en cuenta que:

   - `mutex1` se usa para proteger `queue1`.
   - `mutex2` se usa para proteger `queue2`.
   - minimizar el tiempo que se tiene cogido cada `mutex`.
   - Hay que tratar el caso de que `queue1` y `queue2` esten llenas (esperar a que haya sitio). Existe la función `queue_is_full()` que nos indica si la cola esta llena. Consejo: Usar variables de condición.
   - Hay que tratar el caso de que `queue1` y `queue2` esten vacías (esperar a que haya elementos). Existe la función `queue_is_empty()` que nos indica si la cola esta vacíaa. Consejo: Usar variables de condición.

```
struct queue queue1;
struct queue queue2;
bool queue_is_full(&struct queue queue);
bool queue_is_empty(&struct queue queue);

pthread_mutex_t mutex1;
pthread_cond_t queue1_full = PTHREAD_COND_INITIALIZER;
pthread_cond_t queue1_empty = PTHREAD_COND_INITIALIZER;

pthread_mutex_t mutex2;
pthread_cond_t queue2_full = PTHREAD_COND_INITIALIZER;
pthread_cond_t queue2_empty = PTHREAD_COND_INITIALIZER;

int pthread_cond_signal(pthread_cond_t *cond);
int pthread_cond_wait(pthread_cond_t *restrict cond,
                      pthread_mutex_t *restrict mutex);

int pthread_mutex_lock(pthread_mutex_t *mutex);
int pthread_mutex_trylock(pthread_mutex_t *mutex);
int pthread_mutex_unlock(pthread_mutex_t *mutex);

void *thread1(void *ptr)
{
        while(1) {
                struct element *new;
                struct element *old = remove_element(&queue1);
                new = f1(old);
                insert_element(&queue2, &new);
        }
        return NULL;
}

void *thread2(void *ptr)
{
        while(1) {
                struct element *new;
                struct element *old = remove_element(&queue2);
                new = f1(old);
                insert_element(&queue1, &new);
        }
        return NULL;
}
```