Student Name:
Student ID:

LabReSiD25
Hands-On 13

Università degli Studi di Messina

Condition Variables: Codice d'esempio

```c
// Segnalazione.c

pthread_mutex_t lock;
pthread_cond_t q;
int condizione = 0;

void* thread_attesa(void* arg) {
    pthread_mutex_lock(&lock);
    while (!condizione) {
        pthread_cond_wait(&q, &lock);  // rilascia lock, attende segnale, poi riacquisisce lock
    }
    // condizione vera: fai qualcosa
    pthread_mutex_unlock(&lock);
    return NULL;
}

void* thread_segnalatore(void* arg) {
    pthread_mutex_lock(&lock);
    condizione = 1;
    pthread_cond_signal(&q; // oppure pthread_cond_broadcast(&cond);
    pthread_mutex_unlock(&lock);
    return NULL;
}
```

```c
// ProdCons.c

#define MAX 10
int buffer[MAX];
int count = 0;

pthread_mutex_t lock = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t not_empty = PTHREAD_COND_INITIALIZER;
pthread_cond_t not_full = PTHREAD_COND_INITIALIZER;

void* producer(void* arg) {
    while (1) {
        pthread_mutex_lock(&lock);
        while (count == MAX) {
            pthread_cond_wait(&not_full, &lock);
        }
        buffer[count++] = rand();
```

```
18          pthread_cond_signal(&not_empty);
19          pthread_mutex_unlock(&lock);
20      }
21  }
22
23  void* consumer(void* arg) {
24      while (1) {
25          pthread_mutex_lock(&lock);
26          while (count == 0) {
27              pthread_cond_wait(&not_empty, &lock);
28          }
29          int item = buffer[--count];
30          pthread_cond_signal(&not_full);
31          pthread_mutex_unlock(&lock);
32          // usa item
33      }
34  }
```

Esercizio 1

Si supponga non esista la funziona di libreria `pthread_join()`. Si usino le condition variables per far attendere al padre la terminazione del thread figlio. ATTENZIONE: non è necessario catturare il valore di ritorno del thread figlio.

Esercizio 2

Realizzare, attraverso le condition variables, un meccanismo di barriera che faccia partire "contemporaneamente" 4 threads.