/* link testo http://www.cs.unibo.it/~renzo/so/compiti/2018.06.21.tot.pdf */

Esercizio c1

Per raggiungere un'isola con l'autovettura occorre un traghetto. Il traghetto di questo esercizio ha una sola rampa sulla quale può transitare una automobile alla volta. Il traghetto può imbarcare al massimo MAX automobili. Il traghetto parte solo quando è completamente carico. Occorre disimbarcare tutte le auto prima di imbarcare quelle che desiderano fare la traversata in senso contrario. Le auto possono essere sbarcate in qualsiasi ordine purché una sola alla volta passi sulla rampa. Il processo traghetto è il seguente:

```
traghetto: process:
    while True:
        tmon.al_porto(TERRAFERMA)
        ... naviga
        tmon.al_porto(ISOLA)
        ... naviga
```

Mentre le auto per usare il traghetto chiamano uno dei due segmenti di codice:

```
tmon.imbarca(TERRAFERMA)
    // sulla rampa
tmon.imbarcato(TERRAFERMA)
    // sulla nave
tmon.sbarca(ISOLA)
    // sulla rampa
tmon.sbarcato(ISOLA)
```

```
tmon.imbarca(ISOLA)
        // sulla rampa
tmon.imbarcato(ISOLA)
    // sulla nave
tmon.sbarca(TERRAFERMA)
    // sulla rampa
tmon.sbarcato(TERRAFERMA)
```

Scrivere il monitor tmon

```
#define TERRAFERMA 0
#define ISOLA 1
#define MAX n // numero massimo di navi che possono essere trasportate

monitor tmon {
    /* variabili di stato */
    int direction = [TERRAFERMA, ISOLA];
    int auto[direction];    // auto in attesa in base alla locazione
    int imbarcate;      // auto imbarcate
    int rampa;  // num auto sulla rampa
    int dove_siamo;     // in quale porto ci troviamo

    /* variabili di condizione */
    condition ok2rampa      // la rampa è libera e può essere usata
    condition ok2partire    // la nave è piena e può partire
    condition ok2imbarcare  // la nave è vuota e può imbarcare
    condition ok2sbarcare   // la nave è arrivata e può sbarcare
```

```
tmon(int waiting_car, int direction){
    auto[direction] = waiting_car
    imbarcate = 0
    rampa = 0
    dove_siamo = null
}
/* operazioni */
procedure entry al_porto(int dir) {
    dove_siamo = dir
    if (imbarcati > 0)
        // la nave ha delle macchine sopra
        ok2imbarcare.wait()
    ok2sbarcare.signal()
    if (imbarcati < MAX)
        ok2partire.wait()
    // ok2sbarcare.wait()
}
procedure entry imbarca(int dir) {
    if (imbarcate > MAX && dove_siamo == dir)
        ok2imbarcare.wait()
    // c'è posto sulla barca
    if (rampa != 0)
        ok2rampa.wait() // c'è qualcuno sulla rampa
    // la rampa è libera
    auto[dir]--     // faccio salire una auto sulla rampa
    rampa++
}
procedure entry imbarcato(int dir) {
    rampa--
    imbarcate++
    if (imbarcate < MAX)
        ok2rampa.signal()
    else
        ok2partire.signal()
}
procedure entry sbarca(int dir) {
    if (imbarcate == 0 && dove_siamo == dir)
        ok2sbarcare.wait()
    // la barca è ancora piena
    if (rampa != 0)// c'è qualcuno sulla rampa
        ok2rampa.wait()
    // la rampa è libera
    imbarcate--     // auto esce dalla barca
    rampa++
}
```

```
    procedure entry sbarcato(int dir) {
        rampa--
        if (imbarcate == 0)
            ok2imbarcare.signal()
        else
            ok2rampa.signal()
    }
}
```

Esercizio c2

Facendo uso di semafori ordinari implementare semafori a limite intero N che possano assumere valori -N, -N+1, ...,1, 0, 1, ..., N-1, N. L'invariante dei semafori a limite intero di questo esercizio è: nP - N <= nV + init <= nP + N dove nP è il numero di operazioni P completate, nV il numero delle operazioni V completate e init è il valore iniziale del semaforo.

```
class bounded_semaphore {
    semaphore plus;
    semaphore minus;
    int value;

    bounded_semaphore(int init-val, unsigned max-val){
        value = init-val;
        plus = new semaphore(max-val + init-val);
        minus = new semaphore(max-val - init-val);
    }

    void P(){
        plus.P();
        value--;
        minus.V();
    }

    void V(){
        minus.P();
        value++;
        plus.V();
    }
}
```

Esercizio g1

banchiere multivaluta UNSAFE

Esercizio g2

a) A cosa serve e quando viene eseguito l'algoritmo di calcolo del working set? > L'algoritmo di calcolo del working set serve per evitare il trashing. > Viene evitato segliendo quali pagine di ogni processo vanno mantenute in memoria e quali no. > Questo algoritmo viene eseguito ogni volta che avviene un page fault.

b) Come si calcola la lunghezza massima di un file che si può memorizzare su un file system di tipo fat? >

c) Quali sono le differenze fra un virus e un worm? come ci si difende da questi tipi di malware? >

d) In quali casi la ready queue di uno scheduler può essere vuota? Sono casi fisiologici o patologici della vita di un sistema? >