link testo esame

- Esercizio c1
- Esercizio c2
- Esercizio g1
- Esercizio g2

Esercizio c1

Scrivere il monitor alrv (at least rendez-vous) che fornisce una sola procedure entry: procedure entry void at_least(int n) Quando un processo chiama la funzione at_least vuol dire che vuole sincronizzarsi con un insieme di almeno n processi (incluso il chiamante).

Esempi: - Se un processo chiama at_least(1) non si blocca. - Se il processo p chiama at_least(2) si blocca, - se poi il processo q chiama at_least(2) oppure at_least(1) si sbloccano sia p sia q (la richiesta di p è soddisfatta, ne aspettava almeno 2, p e q) - Se il processo p chiama at_least(2) si blocca, - se poi il processo q chiama at_least(3) si blocca anch'esso perché sebbene la richiesta di p possa essere soddisfatta, q non può ancora sbloccarsi: ci sono solo 2 processi in attesa mentre q ne vuole 3. - Un terzo processo che chiami at_least(x) con x=1,2,3 li sblocca tutti.

Hint: sia w[k] il numero dei processi in attesa di essere in almeno k (quelli che hanno chiamato at_least(k) e non hanno completato l'esecuzione della funzione). Sia s[n]= k=1 n w[k] (rappresenta il numero di processi soddisfacibili: e.g. se ci sono 4 processi in attesa, potrebbero essere soddisfatte le richieste dei processi che ne aspettano almeno 2, almeno 3 o almeno 4). Preso, se esiste, il massimo indice m tale che s[m] m tutti i processi in attesa di essere in n, per n m possono essere sbloccati.

```
#define k       // un qualsiasi numero intero

// considero che tutte le variabili condivise siano inizializzate a 0
monitor alvr{
    int w[k];   // numero dei processi in attesa con richiesta k
    int s[n];   // numero di processi che posso sbloccare con richiesta max n
    condition process;

    procedure entry void at_least(int n){
        // arrivato un nuovo processo, lo aggiungo e salvo i nuovi valori
        w[n]++;
        int index = 0;
        for (int k = 1; k<n; k++){
            s[index] += w[k];
            index++;
            if (index > s.lenght())
```

```
                k = n;  //o break come preferisci, esci dal ciclo
            }
            int m = 0;
            while (m < s.lenght() && s[m] < m){ // trovo l'indice massimo
                m++;
            }
            if (n <= m){
                process.signal();
            }
            process.wait();
            s[n] -= w[n];
            w[n]--;
        }
}
```

Esercizio c2

Dato un servizio di message passing sincrono scrivere, senza fare uso di processi server, un servizio di message passing sincrono concatenato che abbia le seguenti primitive:

```
void chained_send (T msg, list_of_pids dests)
T chained_recv(void)
```

La funzione `chained_send` deve fare in modo che tutti i processi indicati nella lista dests ricevano il messaggio.

Il processo che chiama la `chained_send` si blocca solo fino a quando il primo processo della lista dests non chiama una `chained_recv`, il primo si sblocca quando il secondo chiama la `chained_recv` e così via.

(la funzione `chained_recv` riceve messaggi provenienti da qualsiasi mittente)

Esercizio g1

Esercizio g2