nome e cognome ___________________________ numero di matricola 16 74 __________

CORSO DI SISTEMI OPERATIVI
CORSO DI LAUREA IN INFORMATICA - UNIVERSITA' DI BOLOGNA
SECONDO APPELLO DELLA SESSIONE ESTIVA AA 1999/2000

Esercizio -1. Essersi iscritti per sostenere questa prova.
Esercizio 0. Scrivere correttamente il proprio nome e cognome e numero di matricola in tutti i fogli.

Esercizio 1. Un protocollo di comunicazione affidabile deve consentire a più processi remoti di comunicare fra loro anche in presenza di perdita di messaggi.

```
tsend(m,dest) {send(handler[i],(SND,m,dest))}
trecv(sender) {send(handler[i],(RCV,sender)); recv(handler[i],m);return(m);}
cp[i]: process
       ....
       .... può usare le primitive tsend(m,dest) oppure m=trecv(sender)
       .....
end
```

Posto che il canale di comunicazione possa essere rappresentato come segue:

```
canale: process
       while (true)
       {
              recv(*,(m,sender,dest));
              if (rand() < reliability_ratio)
                     send(dest,(m,sender,dest));
       }
end
```

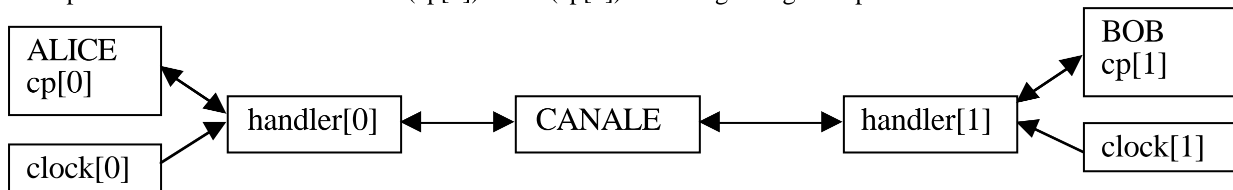
(reliability_ratio è una costante reale nell'intervallo]0,1]);

e che ogni processo abbia a disposizione un processo accessorio clock che spedisce un messaggio tick ogni t millisecondi (tempo normalmente superiore a quello necessario perché il canale consegni un messaggio).

```
clock[i]: process
       while (true)
       {
              sleep(t);
              send(handler[i], (TICK));
       }
```

Scrivere il processo gestore generico handler[i] , usando message passing asincrono, che garantisca una consegna affidabile (rispedisce ogni messaggio fino a quando non è sicuro che sia giunto a destinazione).

Esempio: la comunicazione fra Alice (cp[0]) e Bob(cp[1]) coinvolge i seguenti processi:



Note: l'esempio coinvolge due soli processi ma i programmi devono consentire a molteplici processi di comunicare.
Tutte le comunicazioni sono asincrone, i ritardi sono impredicibili ma in genere inferiori a t millisecondi, ad eccezione della comunicazione fra il clock e l'handler che ha ritardo trascurabile. Non viene richiesto un protocollo efficiente, si può elaborare un messaggio (utente) alla volta. I processi utente non si guastano mai.

nome e cognome ____________________________ numero di matricola 16 74 __________

Esercizio 2a. Ci sono casi di starvation possibili date le specifiche dell'esercizio 1?

Esercizio 2b. Ci sono casi di deadlock?

Esercizio 3. In un distributore di benzina della catena PIGA è in corso la promozione "fai tutto tu".
Il cliente arrivato al distributore deve fare autonomamente benzina o gasolio presso una delle pompe disponibili quindi deve andare alla cassa e pagare (l'importo viene anche segnato su una tesserina di tipo dumbcard, ma questo è irrilevante ai fini dell'esercizio). Le pompe sono tutte moderne e tutte consentono di erogare ogni tipo di carburante.
I processi coinvolti in questo algoritmo concorrente sono:

```
Cliente[i] : process
        while (true) {
                // gira in macchina e... quando sei in riserva
                j=distr.attendipompa();  //il valore di ritorno è il numero della pompa disponibile
                // riempi il serbatoio
                importo=distr.pienofatto(j);
                // vai a pagare
                dist.paga(j,importo);
        }

Pompa[j]: process
        while (true) {
                distr.attendicliente(j);
                // accendi  la pompa
                distr.fineerogazione(j);
                // spegni la pompe misura il carburante
                distr.importo(j,importo);
        }

Cassiere: process
        while (true) {
                distr.incassa(j,importo);
}
```

Regole:

- ogni pompa serve una macchina alla volta
- se tutte le pompe sono occupate il cliente attende (fifo) la liberazione di una qualsiasi pompa
- l'effettiva erogazione (pompa accesa) deve avvenire durante la fase del processo cliente "riempi il serbatoio".
- il cassiere si sincronizza col cliente (paga <-> incassa) e non con la pompa (si fida dell'importo dichiarato dal cliente).

Scrivere il monitor distr.

nome e cognome ___________________________ numero di matricola 16 74 __________

Esercizio4: Uno scienzato pazzo ha deciso di costruire un processore dotato di due istruzioni atomiche
MUL(N) e r=MOD?(N). Queste operazioni operano su una variabile intera globale G di precisione arbitraria (non c'e' overflow) inizializzata a 1 all'atto dell'accensione della macchina e non accedibile s e non tramite queste due funzioni.
Il significato delle due istruzioni atomiche è il seguente:
MUL(N) ::= G=G*N
MOD?(N) ::= if (G mod N == 0) {G=G div N; return 1} else return 0;
Il nostro scienzato afferma che queste due istruzioni sono sufficienti a creare un numero arbitrario di semafori, come? semplicemente associando un numero primo p[i] ad ogni semaforo.
Le istruzioni P e V sarebbero così costruite

```
V(i) {
        MUL(p[i]);
}

P(i) {
        while(! MOD?(p[i]) )
                ;
}
```

L'unico problema, afferma, è che questi semafori non sono fair per il resto funzionano perfettamente.
Ha ragione? Ha torto? perché?