Cognome e nome: _____ Matricola: _____ Posto: ____ ____

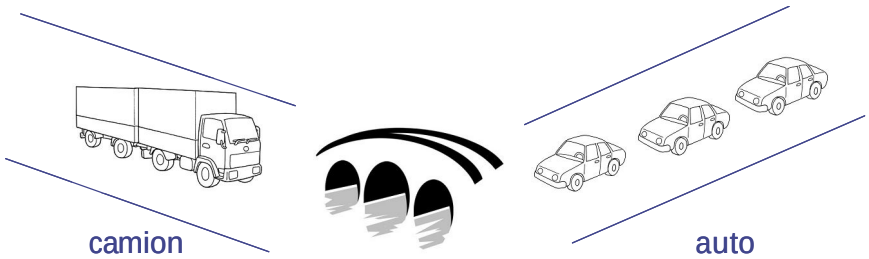
Università degli Studi di Padova - Facoltà di Scienze MM.FF.NN. - Corso di Laurea in Informatica

Regole dell'esame

Il presente esame scritto deve essere svolto in forma individuale in un tempo massimo di 90 minuti dalla sua presentazione.
Non è consentita la consultazione di libri o appunti in forma cartacea o elettronica, né l'uso di palmari e telefoni cellulari.
La correzione e la sessione orale avverranno in data e ora comunicate dal docente durante la prova scritta; i risultati saranno esposti sul sito del docente entro il giorno precedente gli orali.
Per superare l'esame, il candidato deve acquisire almeno 18 punti su tutti i quesiti, inserendo le proprie risposte interamente su questi fogli. Riportare generalità e matricola negli spazi indicati.
Per la convalida e registrazione del voto finale il docente si riserva di proporre al singolo candidato una prova orale.

Quesito 1 – (8 punti):

Si consideri la situazione rappresentata nella figura sottostante dove due strade sono unite da un ponte. Tale ponte è troppo stretto per avere due sensi di marcia: il ponte è dunque a senso unico alternato. Non c'è precedenza prefissata, chi arriva prima comincia, se può, ad attraversare il ponte e fintanto che il ponte non si libera completamente non si può invertire senso di marcia sul ponte.
Si assuma che da un lato del ponte arrivino (per attraversare il ponte) solo camion mentre dall'altro lato arrivino solo auto. Il ponte è sufficientemente robusto da sopportare il passaggio di quante auto si vuole contemporaneamente, ma può sopportare solo un camion alla volta (se un camion sta attraversando il ponte sarà l'unico mezzo sul ponte).

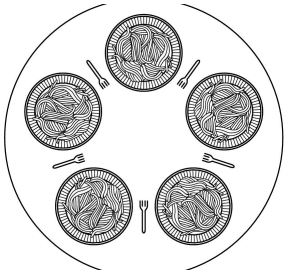


ponte

Si scrivano dei processi che rappresentino i "camion" e dei processi che rappresentino delle "auto" mentre cercano di accedere alla risorsa condivisa "ponte" rispettando le condizioni sopra descritte e utilizzando i ***semafori*** in modo da sincronizzare il sistema senza incorrere in *deadlock*.
Nota: Lo studente si ricordi di inizializzare i valori delle variabili semaforo usate nella sua soluzione.

Cognome e nome: _____ Matricola: _____ Posto: ____ ____

Quesito 2 – (8 punti):



I "filosofi a cena" è un classico problema di sincronizzazione tra più processi (i filosofi) che accedono concorrentemente a risorse condivise (le forchette).

Come visto in aula, lo studente utilizzi i ***semafori*** per scrivere una procedura `Filosofo` che cerchi a fasi alterne di pensare e mangiare. Tali procedure dovranno poter essere eseguite concorrentemente (come fossero un gruppo di filosofi a tavola) evitando *deadlock* del sistema o *starvation* di filosofi.

Si consideri un tavolo con `N` filosofi ed `N` forchette.

Nota: lo studente si ricordi di inizializzare i valori delle variabili semaforo usate nella sua soluzione.

Quesito 3 – (8 punti):
Si ripeta l'esercizio sui "filosofi a cena" risolvendolo però con i ***monitor***.

Cognome e nome: _____ Matricola: _____ Posto: ____ ____

Quesito 4 – (8 punti):

Il problema del "produttore/consumatore" è un classico problema di sincronizzazione tra più processi che accedono concorrentemente a risorse condivise. Lo studente utilizzi i ***monitor*** per scrivere due procedure chiamate `Producer` e `Consumer` che possano essere eseguite concorrentemente al fine di risolvere il problema evitando il *deadlock* del sistema.
(Si consideri il caso in cui le risorse prodotte e non ancora consumate possano essere al massimo `N`).

Cognome e nome: _____ Matricola: _____ Posto: ____ ____

Soluzione

Soluzione al Quesito 1

Il problema è equivalente al problema dei lettori/scrittori. E' dunque corretta una soluzione simile alla seguente dove *db* rappresenta ponte e *rc* rappresenta il numero di lettori:

```
typedef int semaphore;              /* use your imagination */
semaphore mutex = 1;                /* controls access to 'rc' */
semaphore db = 1;                   /* controls access to the database */
int rc = 0;                         /* # of processes reading or wanting to */

void reader(void)
{
     while (TRUE) {                 /* repeat forever */
          down(&mutex);             /* get exclusive access to 'rc' */
          rc = rc + 1;              /* one reader more now */
          if (rc == 1) down(&db);   /* if this is the first reader ... */
          up(&mutex);               /* release exclusive access to 'rc' */
          read_data_base( );        /* access the data */
          down(&mutex);             /* get exclusive access to 'rc' */
          rc = rc – 1;              /* one reader fewer now */
          if (rc == 0) up(&db);     /* if this is the last reader ... */
          up(&mutex);               /* release exclusive access to 'rc' */
          use_data_read( );         /* noncritical region */
     }
}


void writer(void)
{
     while (TRUE) {                 /* repeat forever */
          think_up_data( );         /* noncritical region */
          down(&db);                /* get exclusive access */
          write_data_base( );       /* update the data */
          up(&db);                  /* release exclusive access */
     }
}
```

Soluzione al Quesito 2

Varie soluzioni possibili, ad esempio quella del filosofo mancino:

```
int semaforo f[i] = 1;

Filosofo(i) {
   while(1) {
     <pensa>
     if(i == X) {
        P(f [i+1])%N);
        P(f [i]);
     } else {
        P(f [i]);
        P(f [i+1]%N);
     }
     <mangia>
     V(f [i]);
     V(f [i+1]%N);
   }
}
```

Cognome e nome: _____ Matricola: _____ Posto: ____ ____

Soluzione al Quesito 3

Varie soluzioni possibili, ad esempio:

```
Monitor Tavolo{
   boolean fork_used[5] = false; // forchette numerate da 0 a 4
   condition filosofo[5]; // se lo vogliamo fare in java, questa la dobbiamo
                                  togliere

   raccogli(int n){
      while(fork_used[n] || fork_used[(n+1)%5])
         filosofo[n].wait();
      fork_used[n] = true;
      fork_used[(n+1)%5] = true;
   }
   // in java dovevi aggiungere:
   // (synchronized)
   deposita(int n){
      fork_used[n] = false;
      fork_used[(n+1)%5] = false;
      filosofo[n].notify();   // se lo voglio fare in java devo togliere
                                                queste due "filosofo" e sostituire con
                                                notifyall()
      filosofo[(n+1)%5].notify();
   }
}
Filosofo(i){
   while (true){
      <pensa>
      Tavolo.raccogli(i);
      <mangia>
      Tavolo.deposita(i);
   }
}
```

Soluzione al Quesito 4

Varie soluzioni possibili, ad esempio:

```
monitor ProducerConsumer
      condition full, empty;
      integer count;
      procedure insert(item: integer);
      begin
            if count = N then wait(full);
            insert_item(item);
            count := count + 1;
            if count = 1 then signal(empty)
      end;
      function remove: integer;
      begin
            if count = 0 then wait(empty);
            remove = remove_item;
            count := count – 1;
            if count = N – 1 then signal(full)
      end;
      count := 0;
end monitor;
```

```
procedure producer;
begin
      while true do
      begin
            item = produce_item;
            ProducerConsumer.insert(item)
      end
end;
procedure consumer;
begin
      while true do
      begin
            item = ProducerConsumer.remove;
            consume_item(item)
      end
end;
```

Cognome e nome: _____ Matricola: _____ Posto: ____ ____