Politecnico di Milano
Facoltà di Ingegneria dell'Informazione
Informatica 3
Proff. Campi, Ghezzi, Matera e Morzenti
Prima prova in itinere
10 Maggio 2006

COGNOME E NOME (IN STAMPATELLO)

MATRICOLA

Risolvere i seguenti esercizi, scrivendo le risposte ed eventuali tracce di soluzione negli spazi disponibili.
Non consegnare altri fogli.

Spazio riservato ai docenti

Esercizio 1.
Si consideri il seguente programma C

```
#include <stdio.h>

char GoodPwd[]="MAMMAMIA";

int PwdOK(char * Pwd) {
 gets(Pwd);
 if (strcmp(Pwd, GoodPwd) == 0)
    return 1;
  else
    return 0;
}

void main() {
  char Pwd[8];
  puts ("Inserisci Passwdord:  ");
  if (PwdOK(Pwd))
    puts("Benvenuto nel sistema!");
  else
    puts("Accesso negato");
}
```

Si ricorda che in C le stringhe sono array di caratteri, allocate nello stack e che la funzione strcmp confronta per uguaglianza due stringhe e restituisce zero se le due stringhe sono identiche. Le funzioni gets e puts leggono e stampano stringhe.
Si supponga che la stringa fornita in input al programma a seguito della richiesta di inserimento della password sia più lunga di 8 caratteri. In C cio` non da` luogo ad errore. Cio` pero` puo` causare seri problemi durante l'esecuzione. In particolare, il sottoprogramma PwdOK potrebbe non ritornare all'istruzione di ritorno corretta dell'unita` chiamante. Questo e` infatti la celebre situazione nota come "buffer overflow" che causa seri problemi di sicurezza nelle applicazioni. Si spieghi, con iferimento alla macchina astratta SIMPLESEM perche` puo` accadere che il sottoprogramma non ritorni correttamente al chiamante.

SOLUZIONE
Se la password inserita è troppo lunga può sforare lo spazio allocato per l'array, sovrascrivendo l'indirizzo di ritorno mantenuto nel record di attivazione della funzione PassOK.

Esercizio 2.
Si consideri il seguente programma scritto in un ipotetico linguaggio che adotta regole di visibilita` scope) dinamiche:

```
function main() {
        int X=0;
        B();
        C();
        …
}
function B() {
        int X=100;
        C();
}
function C() {
        print X;
}
```

(a) che cosa viene stampato dal programma?

(b) schizzare lo stato della memoria a run-time dopo la sequenza di chiamate "main chiama B che chiama C", mostrando quale o quali link e` necessario memorizzare nei record di attivazione per supportare la corretta allocazione/deallocazione di memoria e il corretto accesso alle variabili visibili durante l'esecuzione.

(c) il programma sopra presentato sarebbe corretto nel caso il linguaggio adottasse regole di visibilita` statiche?

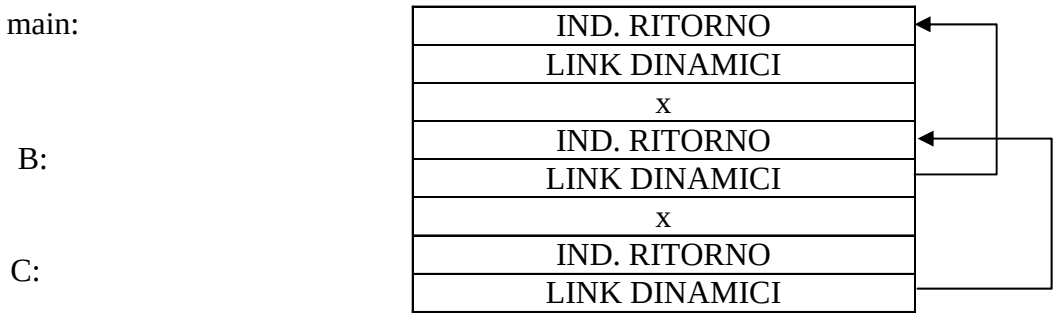
SOLUZIONE

(a)

100

0

(b)



(c) No, perché X non sarebbe visibile in C

Esercizio 3.

I selvaggi di una tribù mangiano servendosi da una pentola comune.
La pentola può contenere P porzioni di cibo (con P non necessariamente maggiore del numero di selvaggi). Ogni selvaggio mangia una porzione per volta. Quando la pentola si svuota (e solo allora), il cuoco provvede a riempirla con nuove P porzioni.

Tratteggiare, in Java e Ada, per le sole parti relative alla sincronizzazione tra i processi, i programmi che realizzano i comportamenti dei selvaggi e del cuoco e la gestione della pentola.
Codificare i moduli di nome **Selvaggio**, **Cuoco**, **Pentola**, e le operazioni **prendiPorzione** e **riempiPentola**.

N.B.: Non è necessario che le soluzioni siano rispettose della sintassi del linguaggio (in particolare, ciò vale per Ada). E' invece importante che la soluzione fornita sia rispettosa della semantica dei costrutti del linguaggio.

SOLUZIONE

JAVA

```
Class Pentola {
   final int P = 40;
   private int nPorzioni;

   public Pentola() {
        nPorzioni = P;
   }

   public syncronized  void prendiPorzione(){
        while ( ! (nPorzioni > 0) )
                try { wait(); }
                catch (InterruptedException e ) {}
        nPorzioni--;
        System.out.println("Pentola: presa porzione");
        notifyAll();
   }

   public syncronized void riempiPentola(){
        while ( ! (nPorzioni = 0) )
                try { wait(); }
                catch (InterruptedException e ) {}
        nPorzioni = P;
        System.out.println("Pentola: riempita di porzioni");
        notifyAll();
     }
}

Class Cuoco extends Thread{
   private Pentola pentola;

    public Cuoco(Pentola p) {
      pentola=p;
   }

   public void run (){
      while(true){
        pentola.riempiPentola();
        System.out.println(“Cuoco: Ho riempito la pentola”);
      }
   }
}
```

```
Class Selvaggio extends Thread{
   private Pentola pentola;

    public Selvaggio(Pentola p) {
      pentola=p;
   }

   public void run (){
      while(true){
        pentola.riempiPentola();
        System.out.println("Selvaggio: Ho preso una porzione");
      }
   }
}

Class Test {
public static void main(String[] args) {
   Pentola p=new Pentola();
   Cuoco c=newCuoco(p)
   Selvaggio s=new Selvaggio(p);
   c.start();
   s.start();
  }
}
```

ADA

```
task Pentola is
        entry prendiPorzione();
        entry riempiPentola();
end Pentola;
task body Pentola is
  nPorzioni: INTEGER := P;
        loop
                select
                        when nPorzioni >= 1
                        accept prendiPorzione() do
                                nPorzioni := nPorzioni – 1;
                        end prendiPorzione;
                or
                        when nPorzioni = 0
                        accept riempiPentola() do
                                nPorzioni := P;
                        end riempiPentola;
                end select;
        end loop;
end Pentola;
```

Esercizio 4.

Considerare il seguente codice Python:

```
 1: a = 0
 2: def f():
 3:    print a
 4: def h(x):
 5:    a = 3 #NB: questo assegnamento di a definisce una nuova variabile locale a
 6:    def k():
 7:        print a
 8:    x()
 9:    x = k
10:    x()
11: g = f
12: h(g)
13: g()
14: print a
```

Simulare l'esecuzione del codice calcolandone l'output.
Per ogni print eseguita indicare il valore stampato, facendo attenzione a riportare il numero di linea della print e a motivare la risposta. **Risposte non motivate non verranno prese in considerazione.**

SOLUZIONE

globale:
linea 1: viene dichiarata una variabile a e viene assegnato ad a un valore intero (0)
linea 11: la funzione f viene memorizzata nella variabile g
linea 12: viene chiamata h(g)
h():
linea 4: h riceve come parameto x la funzione f
linea 5: viene creata una nuova a locale in cui viene memorizzato 3
linea 8: alla linea 8 viene invocato x (quindi f)
f():
linea 3: la prima stampa è 0 perchè la variabile a a cui ci si riferisce in questa linea è globale (python ha scoping statico)
linea 9: al parametro x viene assegnata la funzione k. Questa modifica non influenza la variabile g globale che continua a puntare a f. (pyton usa passaggio parametri per riferimento ma questa non è una modifica all'oggetto ma bensì una modifica alla reference)
linea 10: viene invocata la funzione k
k():
linea 7: la seconda stampa è 3 perche la variabile a cui ci si riferisce è la a di f() in quanto lo scoping è statico e la dichiarazione di k è innestata in f()
linea 13: viene invocata la funzione contenuta nella variabile g che non è stata modificata dalla linea 9. Quindi viene eseguita f()
f():
linea 3: viene stampato 0 (a globale)
linea 14: viene stampato 0 perchè la linea 3 non ha modificato la variabile globale a.