Politecnico di Milano
Facoltà di Ingegneria dell'Informazione
Informatica 3
Proff. Campi, Ghezzi, Matera e Morzenti
Appello del 18 Luglio 2006
Recupero I Parte

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 passaggio di parametri per valore-risultato e in particolare ci si concentri sul passaggio "per risultato". In questo caso, il valore del parametro formale al termine dell'esecuzione del sottoprogramma deve essere ricopiato nel corrispondente parametro attuale. Per effettuare la copia, deve essere noto l'indirisso (l'l-value) di tale parametro.

Esistono casi in cui tale indirizzo potrebbe essere diverso a seconda che esso venga calcolato nel momento della chiamata (e pertanto memorizzato nel record di attivazione) oppure all'atto del ritorno. Si fornisca un semplice esempio che illustri questa possibile differenza.

Soluzione:

```
int a[5] = {0,1,2,3,4};
int b;

f(int p) {
  p=p+b;
  b=b+1;
}

b=3;
f(a[b]);
print a;
```

stampa {0,1,2,6,4} nel caso l-value venga calcolato alla chiamata
stampa {0,1,2,3,6} nel caso l-value venga calcolato al ritorno

Esercizio 2.

Si consideri il seguente problema. Un titolo di borsa ha un valore che puo' essere cambiato mediante un'operazione **deltaVal**(x), dove x e` il valore della variazione del titolo. E` anche possibile che un thread si sottoscriva per essere avvisato quando il valore del titolo scende al di sotto di un valore di interesse, mediante l'operazione **getNotification**(x), dove x in tal caso e` il valore di interesse. Il thread che esegue la getNotification resta sospeso fino a che il valore del titolo non scende al di sotto del valore di interesse, mentre invece prosegue subito se il valore del titolo e` gia` al di sotto della soglia.

Quesito 1

Si tratteggi l'implementazione in Java dei metodi **getNotification** e **deltaVal.**

Quesito 2

Si discuta come si possa realizzare lo stesso comportamento utilizzando ADA. Eventualmente se ne proponga una soluzione che ottiene un effetto simile, anche se con qualche differenza.

```
Class Azione {
  String name;
  float value;

  ...
  public synchronized void deltaVal(float x) {
    value = value – x;
    if (x<0) notifyAll();
  }

  public synchronized void getNotification(float x) {
    while (value >= x) {
      wait();
    }
  }
}
```

ADA
In ada non è possible sospendersi su una condizione di guardia dipendente da un parametro.
Possibili soluzione: rendere la getNotification booleana e non sospensiva, il client deve fare polling.
Eventualmente un oggetto mediatore si incarica di fare polling al posto del client, es:

```
task Poller is
  entry deltaVal(X: in FLOAT)
  entry getNotification(X: in FLOAT)
end Poller;

task body Poller is
  loop
    select
      accept deltaVal(InnerAzione: in AZIONE, X: in FLOAT) do
        InnerAzione.deltaVal(X)
      end deltaVal;
      accept getNotification(InnerAzione: in AZIONE, X: in FLOAT) do
        while (InnerAzione.getNotification(x))
          sleep(TIMEOUT);
      end getNotification;
    end select;
  end loop;
end Poller;
```

Esercizio 3

Si consideri il seguente frammento di programma scritto in un linguaggio ipotetico che consente di dichiarare variabili globali e funzioni all'interno di funzioni e che adotta regole di visibilità (scope) statiche:

```
integer a, b, c; //globali

void h (); //prototipo della funzione, necessario perchè f possa chiamare h

void function f (integer m){ // NB: il parametro m è passato per valore
        integer a, n;

        void g ( ) {
                integer c;
                a = c + m + b; (*)
                h();
        };

        …….
        m = a + n + c; (**)
        ...
        g();
};

void h () {
        integer a, m, c, w;
        …….
        f(c);
        …….
};

……..

void main {
        …….
        h();
        …….
};
```

(a) calcolare l'attributo (distanza, offset) per le variabili che appaiono nell'istruzione (*);
(b) calcolare l'attributo (distanza, offset) per le variabili che appaiono nell'istruzione (**);
(a) schizzare lo stato della memoria (mostrando i links statici e dinamici e la disposizione delle variabili) nel caso in cui main chiami h, che chiama f, che chiama g, che chiama h, che chiama f.

Soluzione

(a) a <1,4> c<0,3> m<1,3> b<2,1>
(b) m<0,3> a<0,4> n<0,5> c<1,2>

CURRENT	0	32	
FREE	1	38	
global	2	a	
	3	b	
	4	c	
Main	5	ret addr	
	6	dyn link	##
	7	static link	2
h()	8	ret addr	
	9	dyn link	5
	10	static link	2
	11	a	
	12	m	
	13	c	
	14	w	
f()	15	ret addr	
	16	dyn link	8
	17	static link	2
	18	m	
	19	a	
	20	n	
g()	21	ret addr	
	22	dyn link	15
	23	static link	15
	24	c	
h()	25	ret addr	
	26	dyn link	21
	27	static link	2
	28	a	
	29	m	
	30	c	
	31	w	
f()	32	ret addr	
	33	dyn link	25
	34	static link	2
	35	m	
	36	a	
	37	n	

Politecnico di Milano
Facoltà di Ingegneria dell'Informazione
Informatica 3
Proff. Campi, Ghezzi, Matera e Morzenti
Appello del 18 Luglio 2006
Recupero II Parte

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.

Dato un albero binario e i tre tipi di visita in Preordine, Inordine e Postordine, si supponga che siano definite tre funzioni, *PRE*, *IN* e *POST*, che restituiscono la posizione relativa di un nodo nell'attraversamento corrispondente. Per esempio, se l'attraversamento Inordine di un albero è ABC e il Preordine è BAC, allora:
IN(A)=1, *IN(B)=2*, *IN(C)=3*
PRE(A)=2, *PRE(B)=1*, *PRE(C)=3*

Quesito 1.
Completare la tabella sotto riportata, indicando se le condizioni indicate in colonna, in un generico albero binario, sono vere per tutte le coppie di nodi *m* e *n* tali che *n* sia antenato di *m* (prima riga) oppure nel caso in cui *m* sia antenato di *n* (seconda riga).
Per esempio, si inserisca il valore "VERO" nell'ultima cella della seconda riga se si ritiene che se *m* è un antenato di *n* allora è vero che *n* precede *m* in un attraversamento in Postordine, o il valore FALSO altrimenti.

	PRE(n) < PRE(m)	**IN(n) < IN(m)**	**POST(n) < POST(m)**
n è antenato di *m*			
m è antenato di *n*			

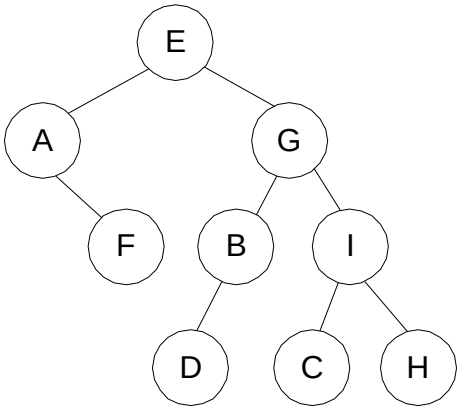
N.B.: un nodo A è un *antenato* di un nodo B se nel cammino dalla radice dell'albero a B si passa attraverso A. Gli antenati sono quindi padri, nonni, ecc.

SOLUZIONE

	PRE(n) < PRE(m)	**IN(n) < IN(m)**	**POST(n) < POST(m)**
n è antenato di *m*	VERO	FALSO	FALSO
m è antenato di *n*	FALSO	FALSO	VERO

Quesito 2.
Si tracci l'albero binario per il quale sono definiti i seguenti attraversamenti:
Preordine: E A F G B D I C H
Inordine: A F E D B G C I H

SOLUZIONE



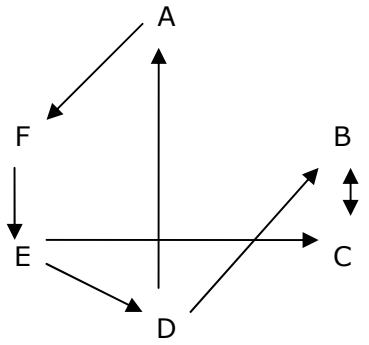
Esercizio 2.
Dato il grafo *G*, rappresentato dalla seguente lista di adiacenza

A -> F
B -> C
C -> B
D -> A -> B
E -> C -> D
F -> E

si risponda ai seguenti quesiti:

1. Disegnare G
2. G è orientato?
3. G è connesso?
4. G è aciclico?
5. Fornire la matrice delle adiacenze di G.

SOLUZIONE



2. G è orientato
3. G è semplicemente connesso, non è fortemente connesso
4. G non è aciclico: contiene il ciclo AFEDA

5. Matrice delle adiacenze

	A	B	C	D	E	F
A	0	0	0	0	0	1
B	0	0	1	0	0	0
C	0	1	0	0	0	0
D	1	1	0	0	0	0
E	0	0	1	1	0	0
F	0	0	0	0	1	0

Esercizio 3

Si tratteggi un algoritmo, basato sul paradigma del *divide et impera*, che, ricevendo come parametri di ingresso due numeri interi positivi *a* ed *n*, calcola la potenza a^n effettuando un numero di operazioni aritmetiche $\theta(\log n)$.
Si argomenti nel modo più preciso possibile che la complessità dell'algoritmo proposto è quella richiesta.

SOLUZIONE

L'algoritmo si basa sulla proprietà

$$a^n = \begin{cases} (a^{n/2})^2 & \text{se n è pari;} \\ (a^{n/2})^2 \cdot a & \text{se n è dispari} \end{cases}$$

```
int  potDivEtImp (int a, int n) {
        int t;
        if (n==1) return a;
        t = potDivEtImp(a, n/2);
        if (n%2 == 1)
                return a * t * t;
        else return t * t;
}
```

Il numero di chiamate ricorsive della funzione è $\theta(\log n)$ perché ad ogni passo di ricorsione il valore dell'esponente viene dimezzato.