Informatica 3 – primo recupero 17 Luglio 2002

Nome _____ **(stampatello)**

Cognome _____ **(stampatello)**

Matr _____

Recupero: ☐ **Prima prova in itinere**

☐ **Seconda prova in itinere**

---

spazio per il docente

Punteggi recupero prima prova	**Punteggi recupero seconda prova**
1) ___/4	**1)** ___/6
2) ___/6	**2)** ___/4
3) ___/5	**3)** ___/4
	4) ___/4

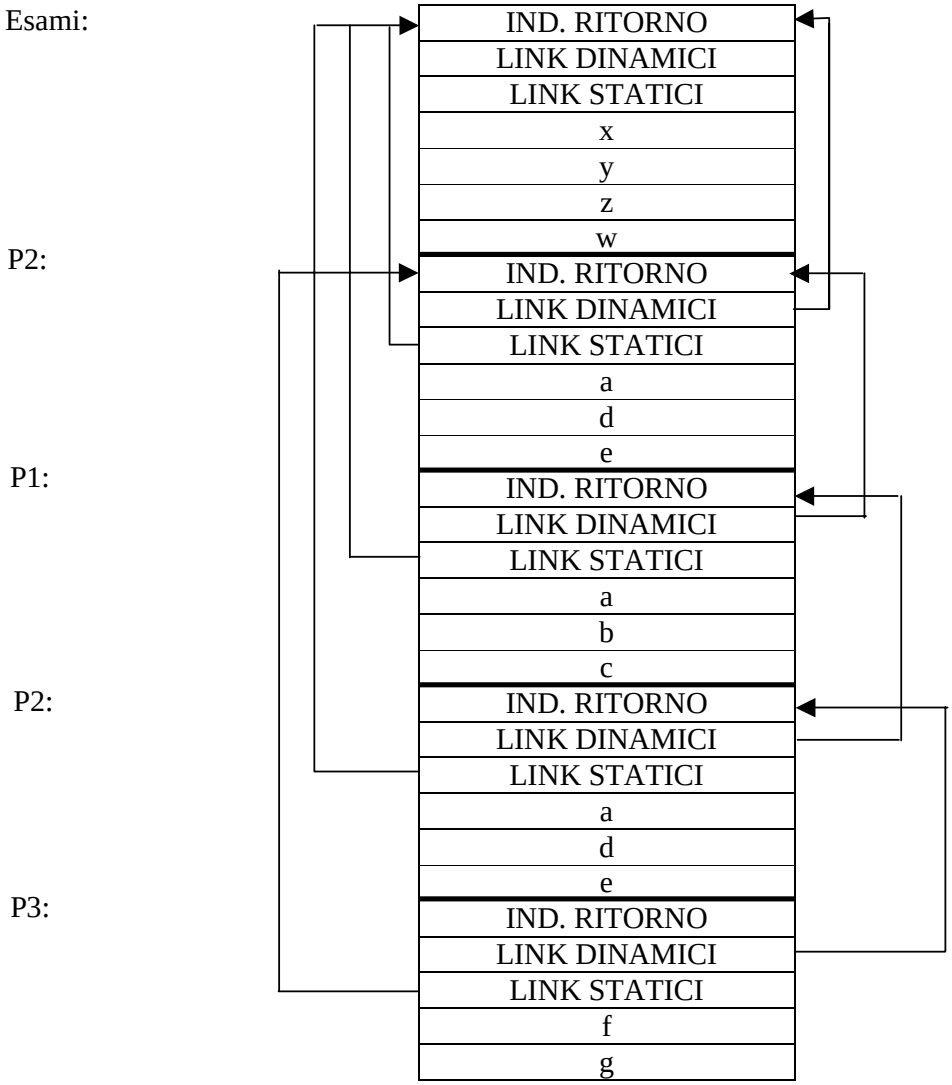
Recupero prima prova in itinere

Esercizio 1 (4 punti)
Si consideri il seguente frammento di programma:

```
program Esame
var x,y,z,w:integer;
        procedure P1()
        var a,b,c:integer;
        ...
        end P1;
        procedure P2()
        var a,d,e: integer;
                procedure P3()
                var f,g:integer;
                z=a+x+g+f+w;          (*)
                …
                end P3;
        end P2;
…
end Esame;
```

1. Mostrare lo stato della macchina astratta dopo la seguente catena di chiamate
   Esame $\rightarrow$ P2 $\rightarrow$ P1 $\rightarrow$ P2 $\rightarrow$ P3
2. Illustrare come staticamente vengono tradotte in coppie <distanza, offset> le variabili presenti nella istruzione contrassegnata da (*)

Sol.



z=a+x+g+f+w
z=<2,5>
a=<1,3>
x=<2,3>
g=<0,4>
f=<0,3>
w=<2,6>

Esercizio 2 (6 punti)
Si consideri il seguente frammento di codice scritto in un linguaggio orientato agli oggetti fortemente tipizzato, con sintassi simile a Java, in cui gli oggetti sono allocati dinamicamente nello heap e valgono le regole di polimorfismo e binding dinamico (si assuma che ciò che non appare dichiarato qui, come le classi a2, a3, b2, b3 e le variabili h1, h2, h3, è dichiarato altrove ed è visibile staticamente al frammento):

```
class a {                       class b extends a {
        int a1;                         int b1;
        a2 f(a3 p) {…}                  b2 f(b3 p) {…}
                                        b3 g(b2 p) { ... }
}                                }
```

```
0. a x = new a;
1. b y = new b; b z = new b;
2. x = z;
3. h1 = x.f(…);
4. h2 = y.f(g(f(...)));
5. z = x;
6. ...
7. x.a1 = y.a1+1;
8. a z = new b;
9. z.b1 = z.a1+5;
```

Si analizzi staticamente il frammento; in particolar modo si evidenzino le istruzioni scorrette dal punto di vista del controllo di tipo o corrette solo sotto certe ipotesi da fare sulle parti di programma qui non mostrate.

Sol.
Ipotesi: metodi e attributi delle classi pubblici per evitare errori in compilazione.
La ridefinizione di f in b deve soddisfare alle regole di covarianza per i risultati e controvarianza per i parametri – in questo caso b2 deve essere un sottotipo di a2, mentre b3 un supertipo di a3 (o =).

0. x e` di tipo a
1. y e z sono di tipo b
2. corretta: assegna un oggetto di un sottotipo ad un di un supertipo – a questo punto x risulta di tipo dinamico b
3. h1 dev'essere di tipo a2 (o supertipo)
4. h2 dev'essere di tipo b2 (o supertipo); scorretta sintatticamente perché tutte le chiamate a metodi di f devono fare riferimento a un oggetto
5. scorretta: non si può assegnare un oggetto di un tipo a una variabile che fa riferimento a un suo sottotipo
6. ...
7. corretta: y e` di un sottotipo di a, dunque a1 e` presente ed e` di tipo int.
8. se e` nello stesso campo d'azione di 1, allora e` sbagliata: ridefinizione di z
9. se la 8 e` corretta, questa e` scorretta: z e` di tipo statico a, supertipo di b – dunque b1 non e` accessibile.

Esercizio 3 (5 punti)
Si definiscano due tipi di thread, detti "a" e "b": il primo si limita a contare da 1 a 5, poi termina, mentre il secondo e` analogo al primo ma conta a rovescio.
Si definisca inoltre un tipo di thread "bx", sottotipo di "b", che:

1) fa partire un thread di tipo "a" e si mette in attesa che questo finisca;
2) si comporta dunque come "b".

Si costruisca infine un programma che lanci un thread di tipo "bx", detto "Y", poi lanci un thread di tipo "a", detto "X", e si metta in attesa della terminazione di "Y".

Sol.

```
public class es1 {
   public static void main (String [] args){
         a x = new a("X");
         bx y = new bx("Y");

         y.start();
         x.start();
         try {
            y.join();
         } catch (InterruptedException ie) {
            System.out.println("Es1 interrotto");
         }
         System.out.println("Fine!");
   }
}


public class a extends Thread{
   private String name;

   public a(String n) {
         this.name = n;
   }

   public void run() {
         System.out.println("sono "+name);
         for(int i=1; i<=5; i++)
            System.out.println(name + ":" + i);
   }
}
public class b extends Thread{
   String name;

   public void b(String n) {
         this.name = n;
   }

   public void run() {
         System.out.println("sono "+name);
         for(int i=5; i<=1; i--)
```

```
            System.out.println(name + ":" + i);
    }
}

public class bx extends b {
    public bx(String n) {
        super.b(n);
    }
    public void run() {
        a ax = new a(name+"-ax");

        ax.start();
        try {
            ax.join();
        } catch (InterruptedException ie) {
            System.out.println("Bx interrotto");
        }
        super.run();
    }
}
```

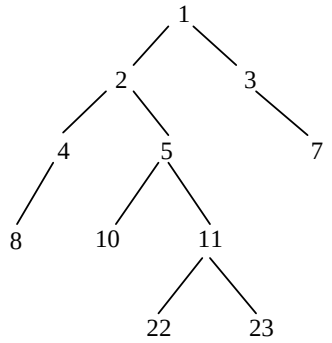
Recupero seconda prova in itinere

Esercizio 1 (6 punti)

In un albero binario la *numerazione per livelli* assegna a ogni nodo *v* dell'albero un valore intero positivo $p(v)$ in accordo alle seguenti regole:

se *v* è la radice	allora	$p(v) = 1$
se *v* è figlio sinistro di *u*	allora	$p(v) = 2 * p(u)$
se *v* è figlio destro di *u*	allora	$p(v) = 2 * p(u) + 1$

In figura è riportato un esempio di tale numerazione.



1 Indicare qual è il valore massimo $p(v)$ che può essere assegnato a un nodo *v* di un albero binario che contiene *n* nodi, giustificando la risposta.
[Suggerimento: considerare, per un albero fissato, qual è il nodo cui viene assegnato il valore massimo.]

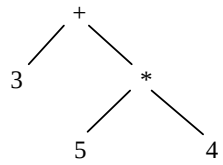
2 Disegnare un albero, con 4 nodi, in cui il limite indicato in precedenza viene raggiunto dal $p(x)$ di un nodo *x* dell'albero.

Sol.

Il valore massimo per p(v) è $2^n - 1$, e viene assegnato alla foglia in un albero degenere lineare con un'unica foglia e in cui tutti i nodi interni hanno solo il figlio destro.

Esercizio 2 (4 punti)

Un albero binario può essere utilizzato per rappresentare un'espressione aritmetica con operatori binari, come mostrato in figura per l'espressione 3+5*4.



La rappresentazione *postfissa* di un'espressione aritmetica è quella in cui, per ogni sottoespressione, si scrivono prima il primo operando, poi il secondo operando, poi l'operatore; nell'esempio citato si ottiene perciò: 3 5 4 * +. Nella rappresentazione *prefissa* si scrive prima l'operatore, poi il primo operando e poi il secondo operando; l'espressione di esempio diventa perciò: + 3 * 5 4.
Implementare due metodi ricorsivi, *postfixPrint* e *prefixPrint* che, ricevendo come parametro un albero che rappresenta un'espressione aritmetica con operatori binari, ne stampino rispettivamente la forma postfissa e prefissa. Si assuma per semplicità che ogni nodo dell'albero contenga un operatore o un operando rappresentato, in ogni caso, come stringa.

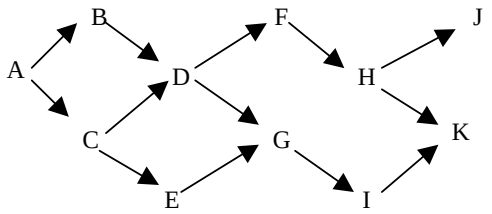
Sol.

```
public class BinNode{
protected Object chiave; // campo chiave
protected BinNode figliosx; // link al figlio sinistro
protected BinNode figliodx; // link al figlio destro
protected BinNode genitore; // link al padre
....
public static void prefixPrint (BinNode radice) {
if(radice==null) return;
else{
        System.out.println(radice.chiave);
        prefixPrint (radice.figliosx);
        prefixPrint (radice. figliodx);
        }
}
public static void postfixPrint (BinNode radice) {
if(radice==null) return;
else{
        postfixPrint (radice.figliosx);
        postfixPrint (radice. figliodx);
        System.out.println(radice.chiave);
        }
}
```

Esercizio 3 (4 punti)

Dato il seguente grafo orientato aciclico, dire quali delle seguenti sequenze di nodi ne costituiscono un ordinamento topologico e quali no, motivando *obbligatoriamente* la risposta per i casi negativi.

A C E B D G F I K H J
A D E C B F H K G I J
A B C E D F H J G I K
A B C D F E G H K J I
A C B E D G I F J H K



Sol.

A C E B D G F I K H J	NO: K precede H
A D E C B F H K G I J	NO: D precede B, E precede C
A B C E D F H J G I K	SI
A B C D F E G H K J I	NO: K precede I,
A C B E D G I F J H K	NO: J precede H

Esercizio 4 (4 punti)

Data una tabella hash di lunghezza m=12, si supponga di dover inserire (in ordine) le chiavi: 7, 24, 32, 36, 41, 19, 21, 90, 102 con la funzione di hash f(k) = k mod m.
Si illustrino i risultati dell'inserimento usando
1 linear probing
2 quadratic probing.

Sol.

Linear Probing

24	36				41	90	7	32	19	21	102

Quadratic Probing

24	36				41	90	7	32	21	102	19