Informatica 3 - secondo recupero 13 Settembre 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) ___/8
2) ___/5	2) ___/4
3) ___/6	3) ___/8

Recupero prima prova in itinere

Esercizio 1 (4 punti)
Nella seguente classe Java, la variabile n ricorre in molti ambiti di visibilità. Quali dichiarazioni di n sono lecite e quali no. Commentare le linee di codice in cui la variabile n appare, evidenziando la dichiarazione di n a cui ci si riferisce.

```
public class X
{
        private int n;                          _____
                                                _____


        public int f()
        {
                int n = 1;                      _____
                return n;                       _____
        }
        public int g(int k)
        {
                int a=0;
                for (int n = 1; n <= k; n++)    _____
                        a = a + n;              _____
                return a;
        }
        public int h(int n)                     _____
        {
                int b=0;
                for (int n = 1; n <= 10; n++)   _____
                        b = b + n;              _____
                return b + n;                   _____
        }
        public int k(int n)                     _____
        {
                if (n < 0)                      _____
                {
                        int k = -n;             _____
                        int n = (int)(Math.sqrt(n));    _____
                        return n;               _____
                }
                else return n;                  _____
        }


        public int m(int k)
```

```
        {
                int a=0;
                for (int n = 1; n <= k; n++)    _____________________________________
                        a = a + n;              _____________________________________
                for (int n = k; n >= 1; n++)    _____________________________________
                        a = a + n;              _____________________________________
                return a;
        }
}
```

Esercizio 2 (5 punti)

Si consideri il seguente programma scritto in un linguaggio immaginario

```
1       program esame()
2       {
3               int x,y;
4               procedure P1 (int z; int w)
5               {
6                       w = 3;
7                       if (y>=2){
8                               w =z + 1;
9                               z = 5;
10                      }
11                      else
12                              w=11;
13              };
14              x=4;
15              y=1;
16              P1(x,y);
17              print(x);       print(y);
18      }
```

Analizzare il valore delle variabili x, y, z w w a seconda che si utilizzi come tipologia di passaggio di parametri:

a) Per riferimento
b) Per valore
c) Per risultato (valore di default 0)

a) Passaggio per riferimento

	x	y	z	w
14				
15				
4				
6				
8				
9				
12				
17				

b) Passaggio per valore

	x	y	z	w
14				
15				
4				
6				
8				
9				
12				
17				

c) Passaggio per risultato

	x	y	z	w
14				
15				
4				
6				
8				
9				
12				
17				

Esercizio 3 (6 punti)

Completare la classe "bx", sottotipo di "b", in modo che faccia partire un thread di tipo "a" e si mette in attesa che questo finisca, segnali l'avvenuta esecuzione del thread precedente, e
attenda 1 s.
Si costruisca poi 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 del segnale da "Y", per poi terminare.
Si scirva poi in Ada lo scheletro di un programma che abbia lo stesso comportamento del codice Java completato precedentemente limitandosi a evidenziare i concetti principali di concorrenza

```
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=5; i>=1; 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=1; i<=5; 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();
        ____________________________________________________
        ____________________________________________________
        ____________________________________________________
        ____________________________________________________
        ____________________________________________________
        ____________________________________________________
        ____________________________________________________
        ____________________________________________________
        ____________________________________________________
        ____________________________________________________
        ____________________________________________________
        ____________________________________________________
        ____________________________________________________
        ____________________________________________________
        ____________________________________________________
        ____________________________________________________
        ____________________________________________________
        ____________________________________________________
        ____________________________________________________
        ____________________________________________________
    }
}
```

Recupero seconda prova in itinere

Esercizio 1 (8 punti)

Si consideri il seguente frammento di codice in Pseudo-C/Java:

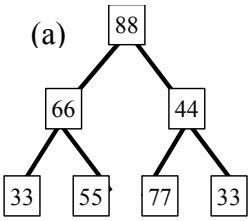
```
j = 1; i = 2;
while ( i <= n)  {
          b[j] = a[i];
           j++;
           i = i*i;
}
```

Si determini l'ordine di grandezza Θ della complessità temporale di tale frammento in funzione di n.

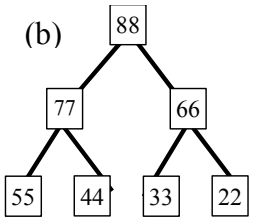
Esercizio 2 (4 punti)

2.1) Quale dei seguenti alberi è uno heap? Per i casi negativi indicare il motivo, per quelli positivi disegnare un array che lo memorizzi.

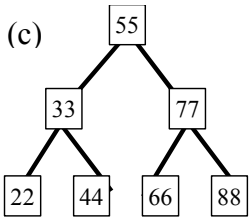
(a)



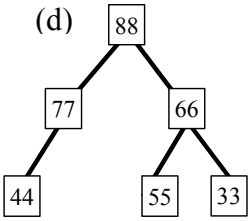
(b)



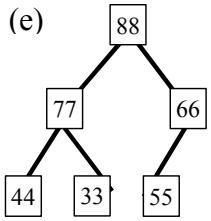
(c)



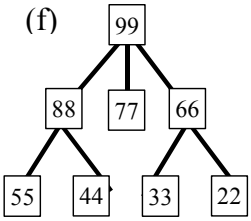
(d)



(e)



(f)



2.2) Quale dei seguenti array memorizza uno heap? Per i casi negativi, indicare il motivo, per quelli positivi disegnare lo heap corrispondente.

(a)

88	66	44	33	55	77	33

(b)

88	77	66	55	44	33	22

(c)

88	44	77	22	33	55	66

(d)

88	77	55	44		33	22

(e)

88	66	77	22	33	44	55

(f)

88	77	22	33	44	55	66

2.3) Visualizzare la costruzione dello heap derivante dall'inserzione dei seguenti valori: 44, 66, 33, 88, 77, 55, 22. Mostrare anche le operazioni di ristrutturazione dell'albero durante le inserzioni.

Esercizio 3 (8 punti)

In un ateneo si tengono le elezioni di 1 rappresentante degli studenti nel consiglio di amministrazione. Ci sono *n* studenti e tutti sono elettori sia attivi (possono votare) sia passivi (possono essere eletti). Ogni studente è identificato dal suo numero di matricola (un valore di tipo int del linguaggio Java). I risultati della votazione sono contenuti in un array di *n* int che contiene tutte le preferenze espresse (si suppone per semplicità che tutti gli studenti abbiano votato esprimendo una preferenza corretta, cioè ogni elemento dell'array sia la matricola di uno studente che è stato votato).
Tratteggiare, in modo informale ma chiaro e inequivoco, un'implementazione il più possibile efficiente in termini di tempo di calcolo, della seguente funzione

```
int   risultatoElezione ( int [] preferenze)
```

che, ricevendo come parametro l'array che contiene le preferenze espresse, restituisce il vincitore delle elezioni, cioè il numero di matricola dello studente che ha ricevuto più preferenze (per semplicità si trascuri la possibilità di "pareggi"). La complessità asintotica dell'algoritmo deve essere dimostrata, anche se non obbligatoriamente in modo del tutto formale.

Soluzione recupero prima prova in itinere

Esercizio 1

```
public class X
{
        private int n;                          CI SI RIFERISCE AD ESSA IN PRESENZA DI ALTRE DICHIARAZIONI
                                                DI VARIABILI DI NOME n MEDIANTE LA ORMA this.n
        public X(int an)
        {
                int n=an;                       IL COSTRUTTORE INIZIALIZZA LA VARIABILE LOCALE N
                                                NON DA' ERRORI DI COMPILAZIONE
        }
        public int f()
        {
                int n = 1;                      NON DA' PROBLEMI, VIENE UTILIZZATA LA VARIABILE LOCALE
                return n;
        }
        public int g(int k)
        {
                int a=0;
                for (int n = 1; n <= k; n++)NON CI SONO PROBLEMI, VIENE UTILIZZATA LA VARIABILE
                                                LOCALE n, CHE ESTENDE LA SUA VISIBILITA' SOLO ALL'INTERNO
                                                DEL CICLO
                        a = a + n;
                return a;
        }
        public int h(int n)
        {
                int b=0;
                for (int n = 1; n <= 10; n++) ERRORE IN COMPILAZIONE! LA VISIBILITA' DEL PARAMETRO n
                                                E DELLA VARIABILE LOCALE n DICHIARATA NEL CICLO SONO
                                                SOVRAPPOSTE.
                        b = b + n;
                return b + n;
        }
        public int k(int n)
        {
                if (n < 0)
                {
                        int k = -n;                     OK

                        int n = (int)(Math.sqrt(n));    ERRORE IN COMPILAZIONE! LA VISIBILITA' DEL P
                                                        PARAMETRO n E DELLA VARIABILE LOCALE n SONO
                                                        SOVRAPPOSTE.
                        return n;
                }
                else return n;
        }
        public int m(int k)
        {
                int a=0;
                for (int n = 1; n <= k; n++)            OK
                        a = a + n;
```

```
            for (int n = k; n >= 1; n++)        OK, LA DICHIARAZIONE DELLA VARIABILE n NON SI
                                                SOVRAPPONE CON QUELLA PRECEDENTE
                a = a + n;
            return a;
        }
}
```

Esercizio 2

a) Passaggio per riferimento

	x	y	z	w
14	4			
15	4	1		
4	4	1	4	1
6	4	3	4	3
8	4	5	4	5
9	5	5	5	5
12	-	-	-	-
17	5	5		

b) Passaggio per valore

	x	y	z	w
14	4			
15	4	1		
4	4	1	4	1
6	4	1	4	3
8	-	-	-	-
9	-	-	-	-
12	4	1	4	11
17	4	1		

c) Passaggio per risultato

	x	y	z	w
14	4			
15	4	1		
4	4	1	0	0
6	4	1	0	3
8	-	-	-	-
9	-	-	-	-
12	4	1	0	11
17	0	11		

.

Esercizio 3

```
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");
          }
          SYNCHRONIZED(THIS) {
             NOTIFY();
          }
          TRY {
             SLEEP(1000);
          } CATCH (INTERRUPTEDEXCEPTION IE) {
             SYSTEM.OUT.PRINTLN("BX INTERROTTO");
          }
…….
   }
}
PUBLIC CLASS ESAME{
   PUBLIC STATIC VOID MAIN (STRING [] ARGS){
          A X = NEW A("X");
          BX Y = NEW BX("Y");

          Y.START();
          X.START();
          TRY {
             SYNCHRONIZED (Y) {
                    Y.WAIT();
             }
          } CATCH (INTERRUPTEDEXCEPTION IE) {
             SYSTEM.OUT.PRINTLN("ES1 INTERROTTO");
          }
          SYSTEM.OUT.PRINTLN("FINE!");
   }
}
```

Ada:

```
procedure MAIN is

task BX;
task A is
          entry ATTENDI
end A;

task body BX is
begin
          A.ATTENDI;
```

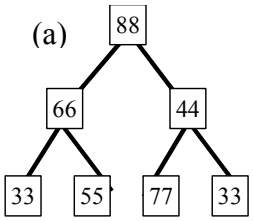
```
        delay 1.0;
        ….
end BX;

task body A is
begin
        accept ATTENDI do
                …….
        end ATTENDI
end A;
```

Recupero seconda prova in itinere

Esercizio 1
$\Theta(\log \log n)$

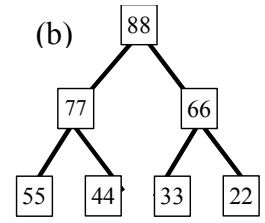
Esercizio 2 2.1

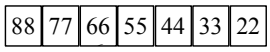
(a)



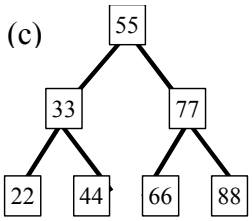
NO: 88-44-77

(b)



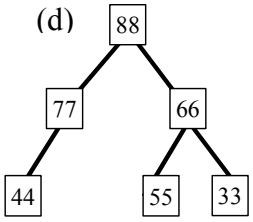


(c)



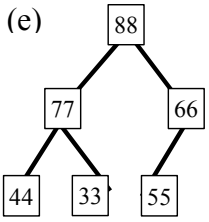
NO: 55-33-44; 55-77-88

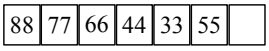
(d)



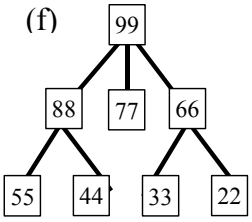
NO: albero non completo

(e)





(f)

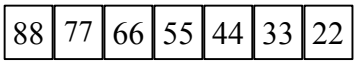


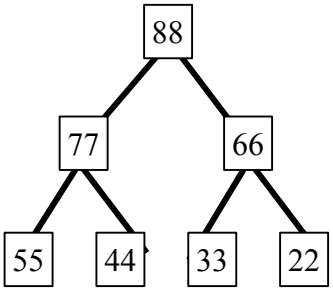
NO: albero non binario

(a)

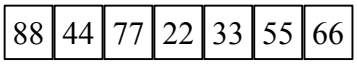
NO: 88-44-55

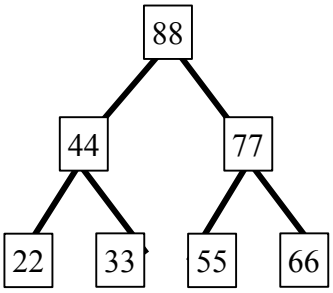
(b)





(c)

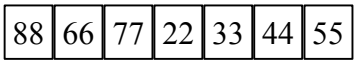


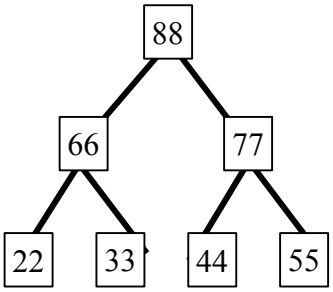


(d)

NO: albero non completo

(e)

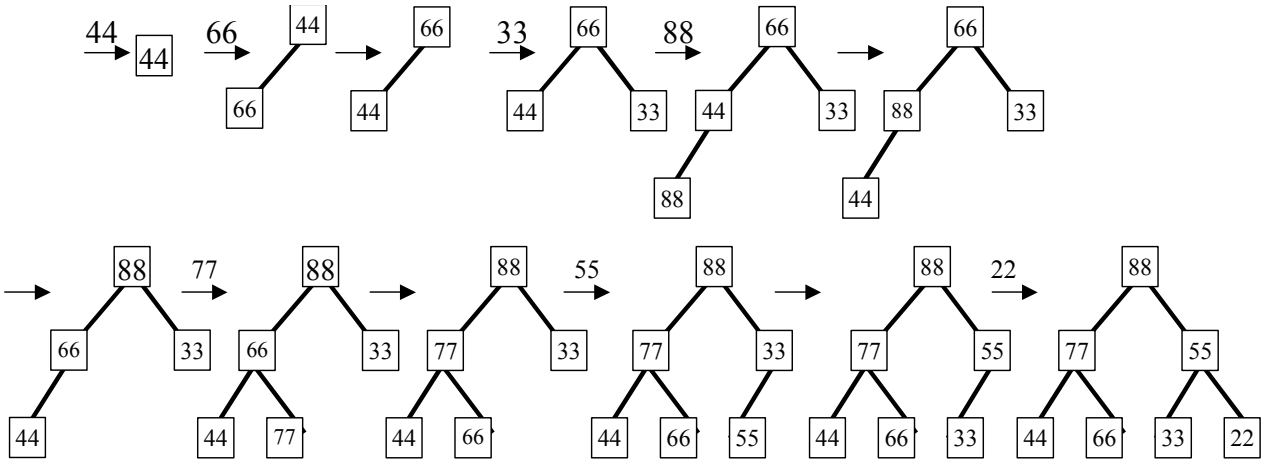




(f)

NO: 88-22-55; 88-22-66

2.3



Esercizio 3

Prima ordinamento dell'array a costo $\Theta(n \log n)$; poi scansione dell'array ordinato per trovare il numero di matricola che ricorre più volte, a costo $\Theta(n)$ perché, essendo l'array ordinato, matricole uguali sono poste in segmenti dell'array, e bastano un contatore e due variabili per la matricola del vincitore e per il numero di ripetizioni. Quindi la complessità totale è $\Theta(n \log n)$.