

Cognome e Nome _____

Matricola _____

Appello del 30 gennaio 2025 in Java

Esercizio 1 [6 punti, da svolgere su questo foglio]

Si considerino le seguenti classi.

```
class A {
    protected int n;

    public A(int n) {
        this.n = n + 1;
    }

    public A(int n, int m) {
        this(n + m);
    }

    1 public int metodo(A a) {
        n = n + a.n;
        return n;
    }
}
```

```
class B extends A {
    public B(int n) {
        this(n, n + 2);
    }

    public B(int n, int m) {
        super(n, m);
    }

    2 public int metodo(A a) {
        n = n + 3 + a.n;
        return n;
    }
}
```

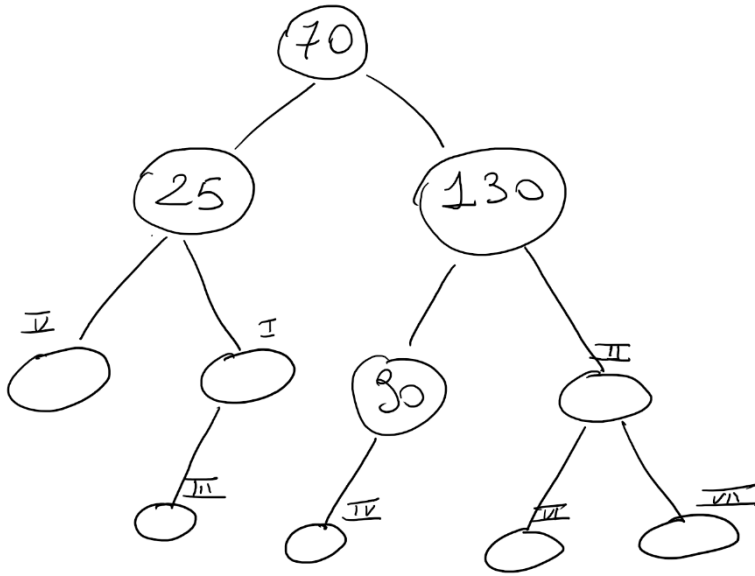
Si dica cosa viene stampato a video dal seguente codice:

```
A a = new A(1, 3);
B b = new B(2);
A ab = new B(4, 5);
System.out.println(a.metodo(ab));
System.out.println(ab.metodo(b));
System.out.println(ab.metodo(ab));
```

Si giustifichi la risposta mostrando in particolare le firme associate a tempo di compilazione e a tempo di esecuzione ad ogni chiamata di metodo:

	Tempo di compilazione	Tempo di esecuzione
a. metodo (ab)		
ab. metodo (b)		
ab. metodo (ab)		

Esercizio 2 [6 punti]



1. [3 punti, da svolgere su questo foglio] Si consideri l'albero binario di ricerca in figura e si fornisca una sequenza di 7 valori interi che, inseriti uno dopo l'altro nell'ABR, vadano a riempire i nodi vuoti nell'ordine indicato dai numeri romani accanto a ciascun nodo (si faccia in modo che nell'albero non ci siano valori duplicati):

--	--	--	--	--	--	--

2. [3 punti] Effettuare una **visita in ampiezza** del grafo corrispondente all'ABR del punto precedente, a partire dal nodo radice 70. A

partire da un **heap di massimo** vuoto, inserire i valori dell'ABR nell'ordine della visita appena fatta, mostrando l'evoluzione passo passo dell'heap.

Esercizio 3 [11 punti]

Si vogliono gestire, in Java, i movimenti associati al credito di SIM di telefonia mobile.

Si assuma, senza scrivere nulla, l'esistenza di una classe astratta **Movimento** con

- una variabile di istanza **idSim** (tipo String, private), che contiene il numero telefonico associato alla SIM a cui il movimento si riferisce;
- una variabile di istanza **tempo** (tipo long, private), che rappresenta l'istante del movimento espresso in secondi trascorsi dalle ore 00:00 del 1° gennaio 2000.

e i seguenti metodi di istanza:

- un costruttore che crea un movimento dati campi **idSim** e **tempo**;
- metodo di accesso **public String getIdSim()**.
- metodo di accesso **public long getTempo()**.
- metodo pubblico astratto **public abstract double getImporto()** che restituisce l'importo in Euro del movimento. In particolare, se il movimento è un movimento di spesa viene restituito un valore minore o uguale a zero (ogni sms costa 0,20 €, mentre ogni chiamata costa 0,15€ di scatto alla risposta più un centesimo per ogni secondo di durata), mentre se è un movimento di ricarica credito viene restituito un valore positivo.

La classe astratta **Movimento** è estesa dalle sottoclassi (non astratte) **Chiamata**, **Sms** e **Ricarica**.

Si tenga conto del fatto che la classe **Chiamata** ha:

- una variabile di istanza **durata** (tipo int, private) che contiene la durata in secondi della chiamata;
- una variabile di istanza **destinatario** (tipo String, private), che contiene il numero chiamato.

La classe **Sms** deve avere:

- una variabile di istanza **destinatario** (tipo String, private), che contiene il destinatario dell'SMS.

La classe **Ricarica** deve avere:

- una variabile di istanza **importo** (tipo double, private), che contiene l'importo della ricarica e deve valere almeno 0,01.

Si assuma, senza scrivere nulla, che tutte le sottoclassi abbiano i costruttori (che prendono in input gli opportuni parametri e che richiamino opportunamente il costruttore della classe **Movimento**), i metodi di accesso per tutti i campi.

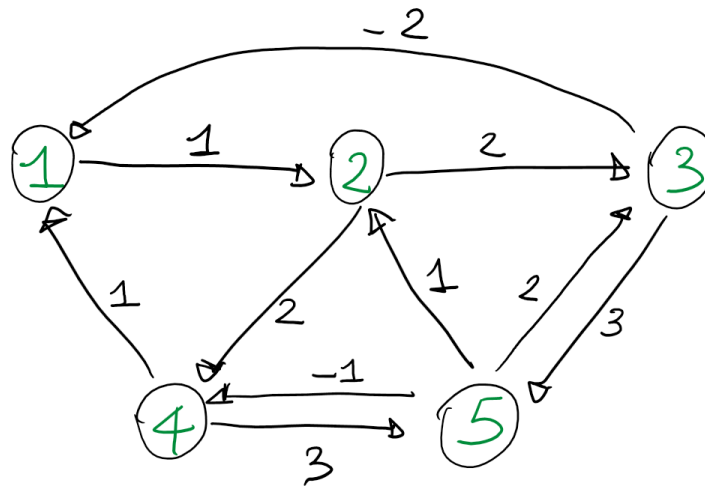
[4 punti] Si scriva l'implementazione del metodo **double getImporto()** per tutte le sottoclassi **Chiamata**, **Sms** e **Ricarica** di **Movimento**.

[7 punti] Si scriva una classe **Gestore** con

- una variabile di istanza **movimenti** (tipo `ArrayList<Movimento>`, private);
- Implementare i seguenti metodi di istanza:
- un costruttore senza parametri che crea un `ArrayList` **movimenti** vuoto.
 - un metodo **public double getCredito (String idSim)** che restituisce il credito della sim **idSim** (ottenuto sommando tutti gli importi dei movimenti nell'`ArrayList` **movimenti** relativi alla sim **idSim**).
 - un metodo **public ArrayList<String> numeriCoivolti ()** che restituisce un `ArrayList` contenente, senza ripetizioni, tutti gli identificativi (i numeri) delle SIM aventi almeno un movimento nell'`ArrayList` **movimenti**, o destinatari di almeno una chiamata o un SMS contenuti nell'`ArrayList` **movimenti**.

Esercizio 4 [11 punti]

Si consideri il grafo (diretto) in figura, per il quale si vogliono calcolare i **cammini minimi tra tutte le coppie di nodi**.



Si elenchino, tra quelli visti a lezione, i possibili algoritmi che possono essere impiegati per risolvere il problema, individuando l'algoritmo che assicura la migliore complessità computazionale nel caso peggiore.

Si applichi quindi tale algoritmo mostrandone una possibile esecuzione **passo-passo**. Ad ogni passo, bisogna mostrare il contenuto di tutte le strutture dati utilizzate dall'algoritmo, compreso il/la vettore/matrice dei padri.

Regole per lo svolgimento della prova scritta:

- Per svolgere il compito si hanno a disposizione **150** minuti.
- Scrivere **subito** nome, cognome, matricola su OGNI FOGLIO (**compreso questo**).
- Durante la prova scritta **non** è possibile abbandonare l'aula.
- Non è ammesso **per nessun motivo** comunicare in qualsiasi modo con altre persone
- È possibile consultare appunti, libri e dispense.
- Qualsiasi strumento elettronico di calcolo o comunicazione (telefoni cellulari, calcolatrici, palmari, computer, etc...) deve essere **completamente disattivato** e **depositato in vista sulla cattedra**
- Mettere in vista sul banco un valido documento di identità.