Cognome e Nome _____

Matricola _____

Programmazione e Algoritmi 2 A.A. 2024/2025

Appello dell'8 luglio 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 + 3;
      }

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

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

1

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

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

      public int metodo(A a) {
            n = n - a.n;
            return n;
      }

}
```

2

Si dica cosa viene stampato a video dal seguente codice:

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

Si consideri la seguente sequenza/array di numeri interi:

[5, 20, 6, 50, 9, 33, 27, 3]

1. **[3 punti]** Mostrare **passo-passo l'evoluzione del min-heap** che si ottiene, a partire dall'heap vuoto, inserendo le chiavi nell'ordine indicato. Dopo aver inserito tutte le chiavi, effettuare **due estrazioni** del minimo mostrando l'heap che si ottiene dopo ogni estrazione.
2. **[3 punti]** Mostrare passo-passo l'esecuzione della procedura **build-max-heap** sull'array indicato.

Esercizio 3 [11 punti]

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

[4 punti] Si scriva 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.
  A questo proposito si tenga conto, per le classi che estendono Movimento, di quanto segue: se il movimento è un movimento di spesa viene restituito un valore minore o uguale a zero (ogni sms costa 0,15 €, mentre ogni chiamata costa 0,20€ di scatto alla risposta più mezzo 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 l'idSim del numero chiamato.

La classe **Sms** deve avere:

- una variabile di istanza **destinatario** (tipo String, private), che contiene l'idSim del 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 le sottoclassi **Chiamata** e **Sms** sono già implementate, con tutti gli opportuni metodi di accesso in lettura alle variabili di istanza.

[4 punti] Si scriva la sottoclasse Ricarica, incluso il costruttore (che prende in input gli opportuni parametri e che richiama opportunamente il costruttore della classe **Movimento**), e metodi di accesso in lettura per tutti i campi, e metodo **`double getImporto()`**.

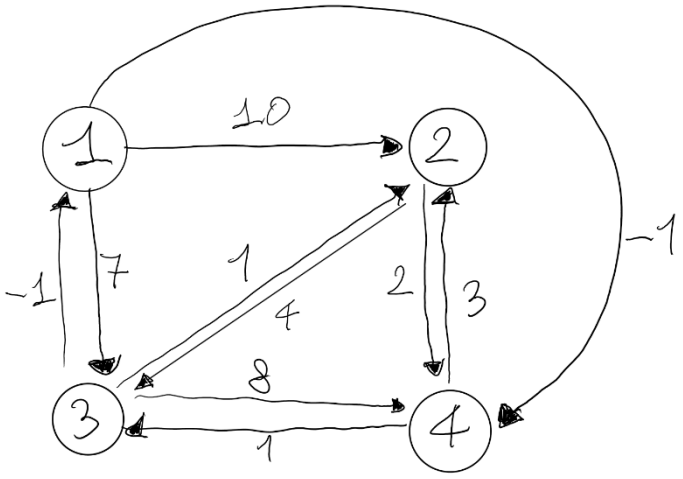
Si assuma, senza scrivere nulla, l'esistenza di una classe **Gestore** con

- una variabile di istanza **movimenti** (tipo ArrayList<Movimento>, private), contenente tutti i movimenti di tutte le sim.

[3 punti] Aggiungere alla classe **Gestore** un metodo **`int getMinutiTraDue (String idSim1, String idSim2)`** che restituisce il tempo totale in minuti nel quale le due sim **idSim1** e **idSim2** sono state impegnate in chiamata (in entrambe le direzioni)**.**

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.**



[1 punto] 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.

[9 punti] 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.

[1 punto] Si descriva il cammino minimo dal nodo **2** al nodo **1**.

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.