Cognome e Nome _____

Matricola _____

Programmazione e Algoritmi 2 A.A. 2022/2023

Appello del 6 settembre 2023

Esercizio 1 [8 punti]

Si considerino le seguenti classi.

```
class A {
      protected int n;

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

      public int metodo(A a) {
            a.n = n + 10;
            return n;
      }

      public int metodo(B b) {
            b.n = n + 20;
            return n;
      }
}
```

```
class B extends A {
      public B(int x, int y) {
            super(x - y);
      }

      public int metodo(A a) {
            a.n = n + 30;
            return n;
      }

      public int metodo(B b) {
            b.n = n + 40;
            return n;
      }
}
```

Si dica cosa viene stampato a video dal seguente codice:

```
A a = new A(2);
B b = new B(8, 4);
A ab = new B(12, 3);
System.out.println(a.metodo(ab));
System.out.println(b.metodo(ab));
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
- l'evoluzione della memoria nelle parti stack ed heap.

Esercizio 2 [9 punti]

Si consideri il seguente array di numeri interi:

[10, 50, 22, 37, 3, 9, 1, 27, 2]

1. **[3 punti]** Mostrare **passo-passo l'esecuzione di Build-max-Heap sull'array**.
2. **[3 punti]** Mostrare **passo-passo l'esecuzione dell'Heap sort** a partire dall'Heap ottenuto al punto 1.
3. **[3 punti]** Mostrare **passo-passo l'Heap di minimo** che si ottiene a partire dall'Heap vuoto inserendo i valori dell'array a partire dalla prima fino all'ultima posizione.

Esercizio 3 [10 punti]

Si vogliono gestire, in Java, gli spettacoli di un cinema/teatro con una sala avente capienza di 524 posti.
Si assuma, senza scrivere nulla, l'esistenza di una classe **Persona** con

- una variabile di istanza **nome** (tipo String, private);
- una variabile di istanza **cognome** (tipo String, private);

e i seguenti metodi di istanza:

- un costruttore che crea un oggetto dati **nome e cognome**;
- metodi pubblici accessori **`getNome()`** e **`getCognome()`**.

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

- una variabile di istanza **spettatori** (tipo Array di Persona, private); si tenga conto non è detto che gli spettatori presenti (laddove siano meno della lunghezza dell'array) siano tutti memorizzati in posizioni consecutive dell'array.
- una variabile di istanza **titolo** (tipo String, private);
- una variabile di istanza **giorno** (tipo int, private), che rappresenta il giorno della data dello spettacolo contando i giorni trascorsi dal 1° gennaio 2000.

e, tra gli altri, i seguenti metodi di istanza:

- un costruttore che crea uno spettacolo dato **titolo** e **giorno**, con l'array di spettatori di lunghezza ***524*** contenente *null* in tutte le posizioni;
- metodo di accesso **`public int getGiorno()`**.
- metodo di accesso **`public String getTitolo()`**.
- Metodo di accesso **`public Persona getSpettatore(int i)`**, che restituisce lo spettatore in posizione **i** (con **i** che può variare da 0 a 523); il metodo restituisce **null** se nella posizione specificata non è presente alcuno spettatore, o se la posizione non è una posizione valida.

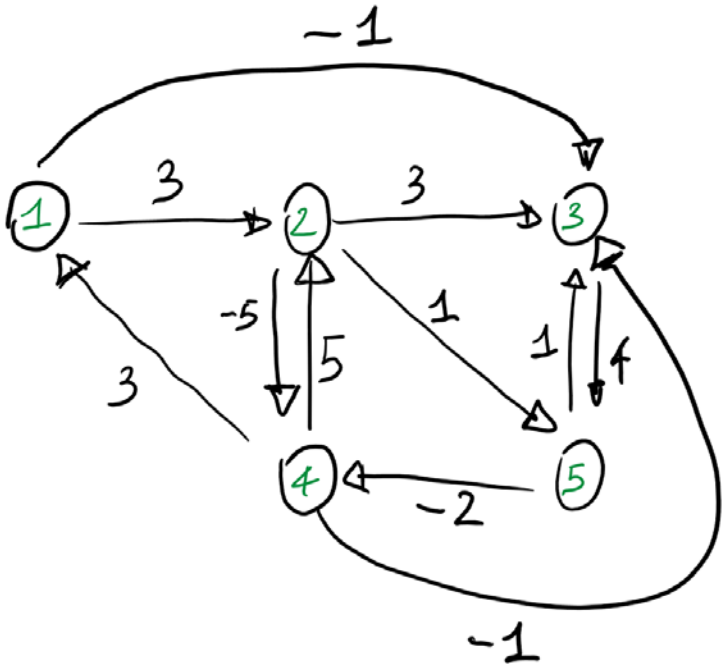
Si assuma, senza scrivere nulla, che la classe astratta **Spettacolo** è estesa dalle sottoclassi (non astratte) **Proiezione** e **Rappresentazione.** Si tenga conto del fatto che la classe **Proiezione** ha una variabile di istanza **durata** (tipo int, private) che contiene la durata in minuti della proiezione, mentre la classe **Rappresentazione** ha una variabile di istanza **cast** (tipo ArrayList<Persona>, private) che contiene la lista degli attori coinvolti nella rappresentazione. Per tutte e due le sottoclassi sono presenti i costruttori (che prendono in input anche la durata e l'ArrayList del cast, rispettivamente) che richiamino opportunamente il costruttore della classe **Spettacolo** e sono presenti metodi di accesso pubblici per le variabili di istanza introdotte (**durata** e **cast**).

[2 punti] Si aggiunga alla classe **Persona** il metodo **`equals(Object o)`** che restituisce *true* se e solo se o è della classe Persona e ha uguale all'oggetto corrente il nome e cognome.

Si scriva una classe **CinemaTeatro** con una variabile di istanza **spettacoli** (tipo ArrayList<Spettacolo>, private);
Implementare i seguenti metodi di istanza:

- **[1 punto]** un costruttore senza parametri che crea un ArrayList **spettacoli** vuoto.
- **[4 punti]** un metodo **`public ArrayList<Spettacoli> estraiSpettacoliSeguiti (Persona p)`** che restituisce un ArrayList contenente tutti e soli gli spettacoli che hanno tra il proprio pubblico la Persona p
- **[3 punti]** un metodo **`public ArrayList<Spettacoli> estraiSpettacoliSeguiti (ArrayList<Persona> a)`** che restituisce un ArrayList contenente, senza ripetizioni, tutti e soli gli spettacoli che hanno tra il proprio pubblico almeno una persona contenuta in **a**.

Esercizio 4 [7 punti]



Utilizzando un algoritmo tra quelli visti a lezione, calcolare cammini minimi a partire dalla sorgente ((**x** mod 5) + 1), dove **x** è l'ultima cifra della propria matricola. In particolare:

a. Giustificare la scelta dell'algoritmo;
b. Mostrare una possibile esecuzione **passo-passo** dell'algoritmo scelto;
c. Dire, giustificando la risposta, se il grafo possiede cicli di peso negativo.

Regole per lo svolgimento della prova scritta:

- Per svolgere il compito si hanno a disposizione **120** 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.