Cognome e Nome _____

Matricola _____

Programmazione e Algoritmi 2 A.A. 2022/2023

Appello straordinario dell'8 maggio 2023

Esercizio 1 [6 punti]

Si considerino le seguenti classi.

```
class A {
      public int metodo(A a)
            return 10;
      }
      public int metodo(B b) {
            return 20;
      }
}
```

```
class B extends A {
      public int metodo(A a) {
            return 30;
      }
      public int metodo(B b) {
            return 40;
      }
}
```

Si dica cosa viene stampato a video dal seguente codice:

```
      A a = new A();
      B b = new B();
      A ab = new B();
      System.out.println(a.metodo(b));
      System.out.println(a.metodo(ab));
      System.out.println(b.metodo(b));
      System.out.println(b.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 di esecuzione ad ogni chiamata di metodo

---

Esercizio 2 [6 punti]

Si consideri la seguente sequenza di numeri interi:

[5, 10, 20, 16, 9, 33, 27, 3, 31]

1. **[3 punti]** Mostrare **passo-passo l'evoluzione del max-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 massimo mostrando l'heap che si ottiene dopo ogni estrazione.
2. **[3 punti]** Mostrare passo-passo l'evoluzione della tabella hash di dimensione 11 (con liste di trabocco), che si ottiene inserendo le chiavi nell'ordine indicato.

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()`**;
- metodo **`equals`** che restituisce *true* se e solo se due oggetti della classe Persona hanno uguale il nome e uguale il cognome.

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

- una variabile di istanza **spettatori** (tipo Array di Persona, private);
- 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()`**.

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**).

Si scriva una classe **CinemaTeatro** con

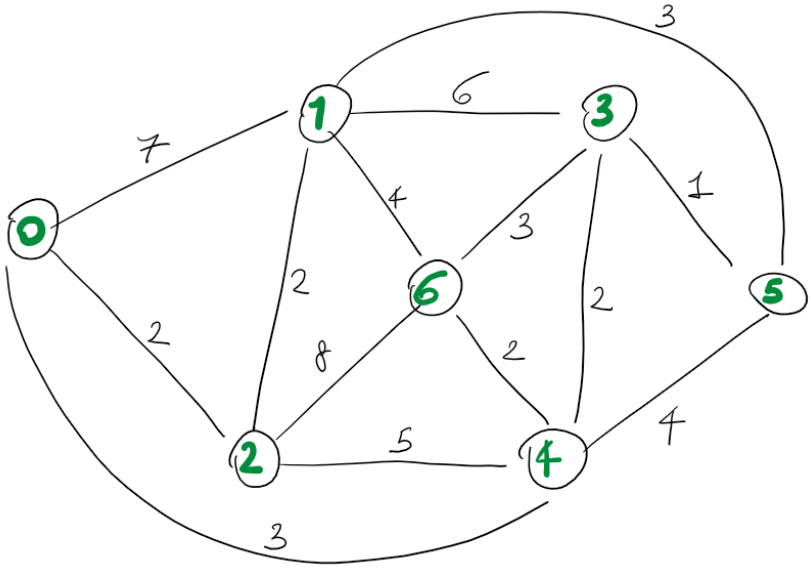
- una variabile di istanza **spettacoli** (tipo ArrayList<Spettacolo>, private);

Implementare i seguenti metodi di istanza:

- un costruttore senza parametri che crea un ArrayList **spettacoli** vuoto.
- un metodo **`public void aggiungiSpettacolo (Spettacolo s)`** che aggiunge lo spettacolo **s** all'ArrayList degli spettacoli.
- un metodo **`public int contaProiezioni ()`** che restituisce il numero di proiezioni contenute nell'ArrayList **spettacoli**.
- un metodo **`public ArrayList<Rappresentazioni> castBy (Persona p)`** che restituisce tutte le rappresentazioni contenute nell'ArrayList **spettacoli** nel cui cast sia presente la **Persona p**.

Esercizio 4 [12 punti]

Si consideri il grafo (non diretto) in figura.



a. **[6 punti]** Mostrare una possibile esecuzione **passo-passo** dell'algoritmo di **Kruskal** per il minimo albero ricoprente, associando ad ogni nodo l'identificativo (intero) della componente connessa in cui si trova e facendo vedere, ad ogni passo, l'evoluzione dell'array in cui in posizione i c'è l'intero della componente connessa associata al nodo i.

b. **[6 punti]** Mostrare una possibile esecuzione **passo-passo** dell'algoritmo di **Prim** per il minimo albero ricoprente, facendo partire l'algoritmo dal nodo ((**x+2**) mod 7), dove **x** è l'ultima cifra della propria matricola. Ad ogni passo, bisogna mostrare il contenuto della coda con priorità utilizzata dall'algoritmo e il vettore dei padri.

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.