Cognome e Nome _____

Matricola _____

Programmazione e Algoritmi 2
A.A. 2024/2025

Appello del 15 gennaio 2025

COMPITO N° 1

PRIMA PARTE

Esercizio 1 [10 punti]

Si consideri il seguente codice Python.

```
class A:
    @property
    def valore (self) -> int:
        return 10

class B:
    @property
    def valore (self) -> int:
        return 20

class C (A, B):
    pass
```

```
class D (B, A):
    pass

class E (C, B):
    pass

c = C()
print (c.valore)

d = D()
print (d.valore)

e = E()
print (e.valore)
```

Denotando la classe `object` con O, si calcoli il MRO di tutte le classi, e si scriva in tabella il procedimento e il risultato finale:

classe	**MRO**
A	L(A) = A + merge (L(O), O) = A + merge (O, O) = AO
B	
C	
D	
E	

Se qualcuno di questi MRO non esiste, si eliminino idealmente le linee di codice relativo (sia della definizione che della successiva creazione di oggetti e invocazione di metodo) e si scriva nel riquadro a destra cosa viene stampato a video dal codice restante.

Esercizio 2 [12 punti]

Si consideri la seguente sequenza di numeri interi:

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

1. **[6 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. **[6 punti]** Mostrare passo-passo l'evoluzione della tabella hash di dimensione 11 (con hashing doppio), che si ottiene inserendo le chiavi nell'ordine indicato.

Esercizio 3 [12 punti]

Si implementino in Python le classi **Autore** e **Libro**.
Attenzione: Tutto il codice deve essere annotato con i tipi opportuni.

[4 punti] La *classe* **Autore** ha i seguenti attributi di istanza:

- **_codice** (un **intero**)
- **_nome** (una **stringa**)
- **_cognome** (una **stringa**)

ed i seguenti metodi di istanza:

- *__init__* che inizializza un oggetto della *classe* **Autore** assegnando codice, nome e cognome
- proprietà (`@property`) in lettura per tutti gli attributi, con lo stesso nome (senza _);
- metodo **__eq__ (self, o:object)** che restituisce **True** se **o** è un autore con lo stesso codice di self, (indipendentemente da nome e cognome), **False** altrimenti.
- metodo **__str__ (self)** che restituisce la rappresentazione come stringa di un autore, nella forma `Autore n m, codice c`, dove n, m e c rappresentano il nome, il cognome e il codice dell'autore **self**, rispettivamente.

[4 punti] La *classe* **Libro** ha i seguenti attributi di istanza:

- **_codice** (un **intero**)
- **_titolo** (una **stringa**)
- **_autori** (una **list** di **Autore**)

ed i seguenti metodi di istanza:

- *__init__* che inizializza un oggetto della *classe* **Libro** assegnando codice e titolo (e creando una lista vuota di autori)
- metodo **add_autore (self, Autore a)** che aggiunge **a** alla lista degli autori del libro, qualora non sia già presente in essa.
- metodo **__eq__ (self, o:object)** che restituisce **True** se **o** è un libro con lo stesso codice di self, (indipendentemente dagli altri campi), **False** altrimenti.
- metodo **__str__ (self)** che restituisce la rappresentazione come stringa di un libro, nella forma `Libro t, codice c`, dove t e c rappresentano il titolo e il codice del libro **self**, rispettivamente.

[4 punti] Scrivere infine, fuori dalle classi, un metodo **filtra_libri (libri:list[Libro], autori:list[Autore]) -> list[Libro]** che restituisce una nuova lista contenente tutti e soli i libri della lista **libri** aventi tutti gli autori presenti nella lista **autori**.

SECONDA PARTE

Esercizio 4 [12 punti]

Per chi svolge solo la seconda parte: tenere presente, senza svolgerlo, la traccia dell'esercizio 3.

Attenzione: Tutto il codice deve essere annotato con i tipi opportuni.

[5 punti] Si implementi la *classe* **LibroFasciaEtà** estende la classe **Libro** ed ha i seguenti attributi di istanza:

- **_min_age** (un **intero**)
- **_max_age** (un **intero**)

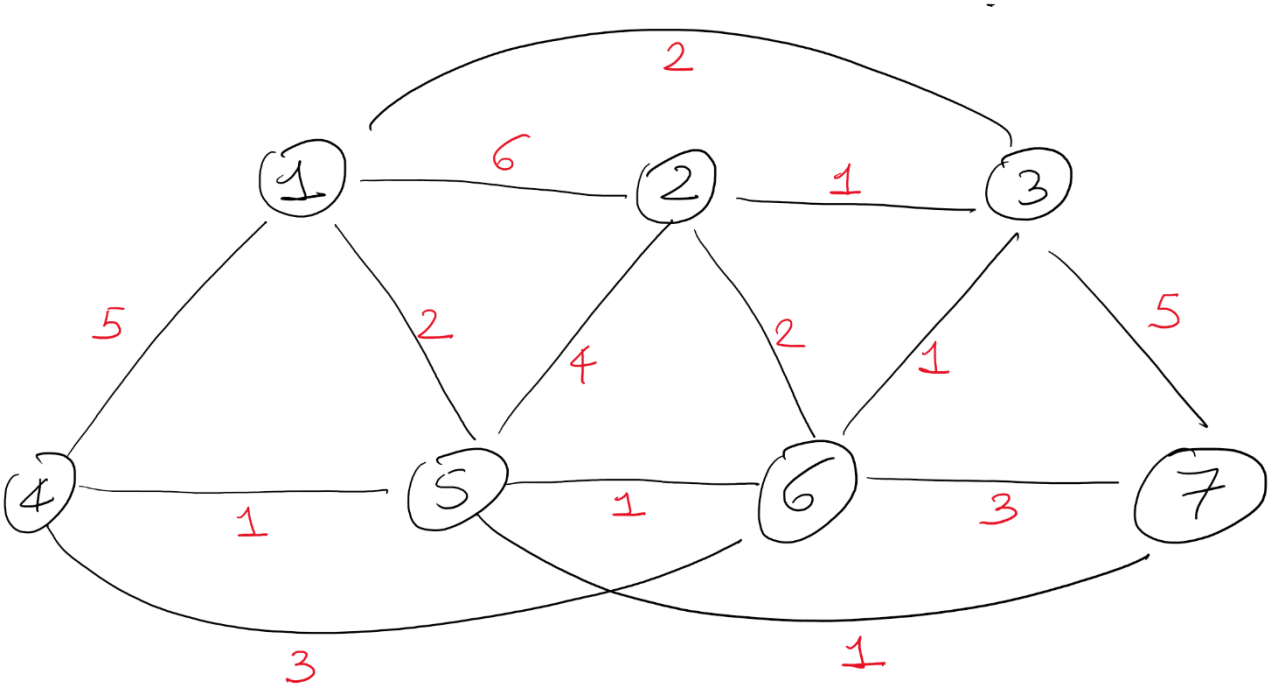
ed i seguenti metodi di istanza:

- *__init__* con tutti i parametri necessari, che inizializza un oggetto della *classe* **LibroFasciaEtà** assegnando **_min_age** e **_max_age** (e richiamando opportunamente il costruttore della superclasse)
- metodo **is_suitable (self, age:int) -> bool** che restituisce **True** se **self** è un libro adatto per l'età **age**, cioè se age è compreso tra **_min_age** e **_max_age** estremi inclusi, **False** altrimenti.
- metodo **__str__ (self)** che sftuttando il metodo omonimo della superclasse restituisce la rappresentazione come stringa di un libro, nella forma
  `Libro t, codice c, adatto da min a max anni`, dove t, c, min e max rappresentano il titolo, il codice, l'età minima e massima consigliata del libro **self**, rispettivamente.

[7 punti] Scrivere infine, fuori dalle classi, un metodo **autore_più_presente (libri:list[Libro]) -> Autore** che, restituisce l'autore (o uno degli autori) che ha scritto il maggior numero di libri. Si analizzi la complessità computazionale del metodo proposto, in funzione del numero n di libri e del numero m di autori (assumendo che ogni libro abbia un numero O(1) di autori).
Se necessario, si aggiungano alle classi dell'**Esercizio 3** tutti i metodi necessari per far funzionare correttamente il codice e si discuta quanto fatto.
Suggerimento: si usi un dizionario per memorizzare il numero di libri scritti da ogni autore.

Esercizio 5 [22 punti]

Si consideri il grafo diretto in figura.



a. **[11 punti]** Mostrare una possibile esecuzione **passo-passo** dell'algoritmo di **Dijkstra** per i cammini minimi da singola sorgente, a partire dal nodo sorgente **4**. Ad ogni passo, bisogna mostrare il contenuto della coda con priorità utilizzata dall'algoritmo e del vettore dei padri.
Si descriva il cammino minimo dal nodo 4 al nodo 2. Che peso ha?

b. **[11 punti]** Mostrare una possibile esecuzione **passo-passo** dell'algoritmo di **Prim** per il minimo albero ricoprente, a partire dal nodo sorgente **2**. Ad ogni passo, bisogna mostrare il contenuto della coda con priorità utilizzata dall'algoritmo e del vettore dei padri.
Che peso complessivo ha il minimo albero ricoprente?

Regole per lo svolgimento della prova scritta:

- Per svolgere il compito si hanno a disposizione **100** minuti per chi svolge solo la seconda parte e **180** minuti per chi svolge il compito totale (chi non consegna dopo 100 minuti automaticamente rinuncia al voto dell'esonero).
- Scrivere **subito** nome, cognome, matricola su OGNI FOGLIO (**compresi quelli della traccia, negli appositi spazi**).
- 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 lontano dalla vostra postazione.**
- Mettere in vista sul banco un valido documento di identità.

Cognome e Nome _____

Matricola _____

COMPITO N° 1

Cognome e Nome _____

Matricola _____

Programmazione e Algoritmi 2
A.A. 2024/2025

Appello del 15 gennaio 2025

COMPITO N° 2

PRIMA PARTE

Esercizio 1 [10 punti]

Si consideri il seguente codice Python.

```
class A:
    @property
    def valore (self) -> int:
        return 15

class B:
    @property
    def valore (self) -> int:
        return 25

class C (B, A):
    pass
```

```
class D (A, B):
    pass

class E (C, A):
    pass

c = C()
print (c.valore)

d = D()
print (d.valore)

e = E()
print (e.valore)
```

Denotando la classe `object` con O, si calcoli il MRO di tutte le classi, e si scriva in tabella il procedimento e il risultato finale:

classe	MRO
A	L(A) = A + merge (L(O), O) = A + merge (O, O) = AO
B	
C	
D	
E	

Se qualcuno di questi MRO non esiste, si eliminino idealmente le linee di codice relativo (sia della definizione che della successiva creazione di oggetti e invocazione di metodo) e si scriva nel riquadro a destra cosa viene stampato a video dal codice restante.

Esercizio 2 [12 punti]

Si consideri la seguente sequenza di numeri interi:

[10, 2, 16, 5, 90, 32, 27, 3, 31]

3. **[6 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.
4. **[6 punti]** Mostrare passo-passo l'evoluzione della tabella hash di dimensione 11 (con hashing doppio), che si ottiene inserendo le chiavi nell'ordine indicato.

Esercizio 3 [12 punti]

Si implementino in Python le classi **Autore** e **Libro**.
Attenzione: Tutto il codice deve essere annotato con i tipi opportuni.

[4 punti] La *classe* **Autore** ha i seguenti attributi di istanza:

- **_codice** (un **intero**)
- **_nome** (una **stringa**)
- **_cognome** (una **stringa**)

ed i seguenti metodi di istanza:

- *__init__* che inizializza un oggetto della *classe* **Autore** assegnando codice, nome e cognome
- proprietà (`@property`) in lettura per tutti gli attributi, con lo stesso nome (senza _);
- metodo **__eq__ (self, o:object)** che restituisce **True** se **o** è un autore con lo stesso codice di self, (indipendentemente da nome e cognome), **False** altrimenti.
- metodo **__str__ (self)** che restituisce la rappresentazione come stringa di un autore, nella forma `Autore n m, codice c`, dove n, m e c rappresentano il nome, il cognome e il codice dell'autore **self**, rispettivamente.

[4 punti] La *classe* **Libro** ha i seguenti attributi di istanza:

- **_codice** (un **intero**)
- **_titolo** (una **stringa**)
- **_autori** (una **list** di **Autore**)

ed i seguenti metodi di istanza:

- *__init__* che inizializza un oggetto della *classe* **Libro** assegnando codice e titolo (e creando una lista vuota di autori)
- metodo **add_autore (self, Autore a)** che aggiunge **a** alla lista degli autori del libro, qualora non sia già presente in essa.
- metodo **__eq__ (self, o:object)** che restituisce **True** se **o** è un libro con lo stesso codice di self, (indipendentemente dagli altri campi), **False** altrimenti.
- metodo **__str__ (self)** che restituisce la rappresentazione come stringa di un libro, nella forma `Libro t, codice c`, dove t e c rappresentano il titolo e il codice del libro **self**, rispettivamente.

[4 punti] Scrivere infine, fuori dalle classi, un metodo **filtra_libri (libri:list[Libro], autori:list[Autore]) -> list[Libro]** che restituisce una nuova lista contenente tutti e soli i libri della lista **libri** aventi tutti gli autori presenti nella lista **autori**.

SECONDA PARTE

Esercizio 4 [12 punti]

Per chi svolge solo la seconda parte: tenere presente, senza svolgerlo, la traccia dell'esercizio 3.

Attenzione: Tutto il codice deve essere annotato con i tipi opportuni.

[5 punti] Si implementi la *classe* **LibroFasciaEtà** estende la classe **Libro** ed ha i seguenti attributi di istanza:

- **_min_age** (un **intero**)
- **_max_age** (un **intero**)

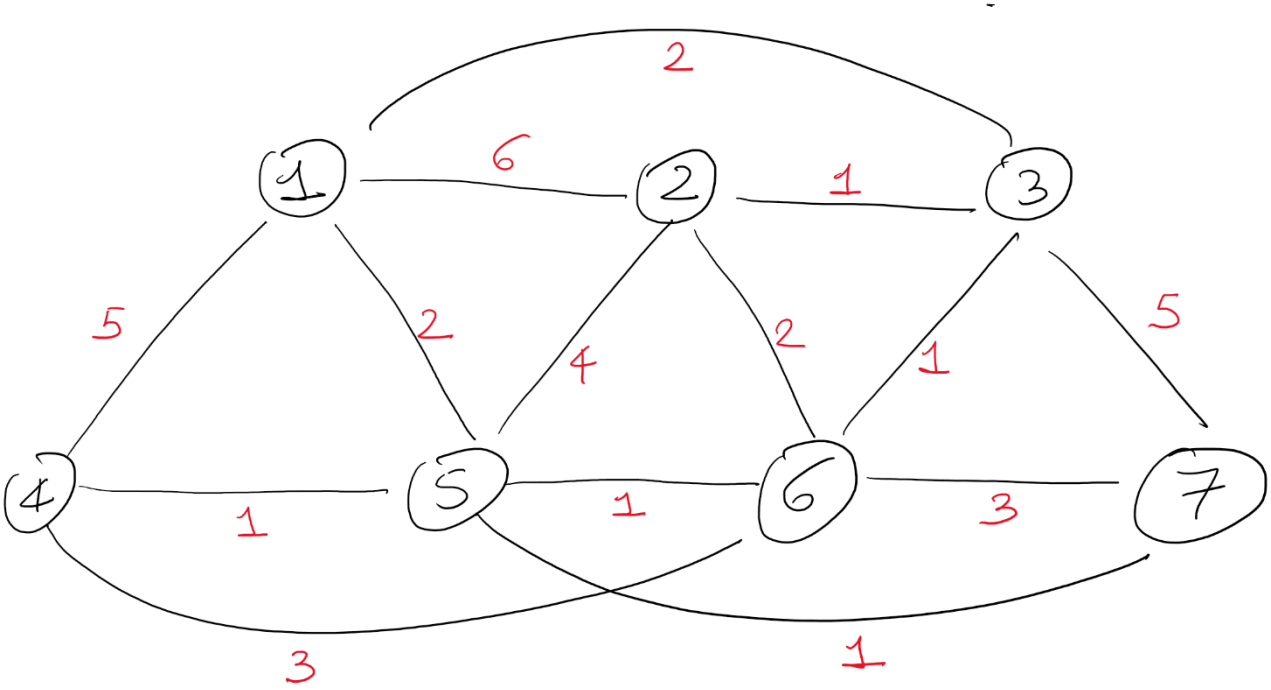
ed i seguenti metodi di istanza:

- *__init__* con tutti i parametri necessari, che inizializza un oggetto della *classe* **LibroFasciaEtà** assegnando **_min_age** e **_max_age** (e richiamando opportunamente il costruttore della superclasse)
- metodo **is_suitable (self, age:int) -> bool** che restituisce **True** se **self** è un libro adatto per l'età **age**, cioè se age è compreso tra **_min_age** e **_max_age** estremi inclusi, **False** altrimenti.
- metodo **__str__ (self)** che sftuttando il metodo omonimo della superclasse restituisce la rappresentazione come stringa di un libro, nella forma
  `Libro t, codice c, adatto da min a max anni`, dove t, c, min e max rappresentano il titolo, il codice, l'età minima e massima consigliata del libro **self**, rispettivamente.

[7 punti] Scrivere infine, fuori dalle classi, un metodo **autore_più_presente (libri:list[Libro]) -> Autore** che, restituisce l'autore (o uno degli autori) che ha scritto il maggior numero di libri. Si analizzi la complessità computazionale del metodo proposto, in funzione del numero n di libri e del numero m di autori (assumendo che ogni libro abbia un numero O(1) di autori).
Se necessario, si aggiungano alle classi dell'**Esercizio 3** tutti i metodi necessari per far funzionare correttamente il codice e si discuta quanto fatto.
Suggerimento: si usi un dizionario per memorizzare il numero di libri scritti da ogni autore.

Esercizio 5 [22 punti]

Si consideri il grafo diretto in figura.



c. **[11 punti]** Mostrare una possibile esecuzione **passo-passo** dell'algoritmo di **Dijkstra** per i cammini minimi da singola sorgente, a partire dal nodo sorgente **7**. Ad ogni passo, bisogna mostrare il contenuto della coda con priorità utilizzata dall'algoritmo e del vettore dei padri.
Si descriva il cammino minimo dal nodo 7 al nodo 3. Che peso ha?

d. **[11 punti]** Mostrare una possibile esecuzione **passo-passo** dell'algoritmo di **Prim** per il minimo albero ricoprente, a partire dal nodo sorgente **3**. Ad ogni passo, bisogna mostrare il contenuto della coda con priorità utilizzata dall'algoritmo e del vettore dei padri.
Che peso complessivo ha il minimo albero ricoprente?

Regole per lo svolgimento della prova scritta:

- Per svolgere il compito si hanno a disposizione **100** minuti per chi svolge solo la seconda parte e **180** minuti per chi svolge il compito totale (chi non consegna dopo 100 minuti automaticamente rinuncia al voto dell'esonero).
- Scrivere **subito** nome, cognome, matricola su OGNI FOGLIO (**compresi quelli della traccia, negli appositi spazi**).
- 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 lontano dalla vostra postazione.**
- Mettere in vista sul banco un valido documento di identità.

Cognome e Nome _____

Matricola _____

COMPITO N° 2