Cognome e Nome _____

Matricola _____

Programmazione e Algoritmi 2 A.A. 2025/2026

Esonero del 17 novembre 2025

Compito n° 1

Esercizio 1 [10 punti]

Si consideri il seguente codice Python.

```
class A:
    def __eq__(self, o: object) -> bool:
        print ("eq di A")
        if isinstance(o, A): return True
        return NotImplemented

class B:
    def __eq__(self, o: object) -> bool:
        print ("eq di B")
        if isinstance(o, B): return True
        if isinstance(o, A): return True
        return NotImplemented

class C (B):
    pass
```

```
class D (A):
    pass

class E (C, A):
    pass

class F (B, E):
    pass

c = C()
d = D()
e = E()
f = F()

print (f"d==c is {d==c}")
print (f"c==d is {c==d}")
```

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

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

Si scriva nel riquadro a destra cosa **esclusivamente** ciò che viene stampato a video dal codice.

Nel caso in cui il codice dia errore in qualche punto (poiché ad esempio non è possibile calcolare qualche MRO), escludere dal codice il numero minimo di linee in modo da non ottenere errori, commentandole con **#** sul listato stampato in alto su questo foglio, e quindi stampare cosa risulta dal nuovo codice così ottenuto.

Esercizio 2 [10 punti]

Si consideri la seguente sequenza/array di numeri interi:

[2, 48, 9, 27, 11, 33, 18, 62, 10]

1. **[5 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. **[5 punti]** Mostrare passo-passo l'esecuzione della procedura **build-max-heap** sull'array indicato.

Esercizio 3 [14 punti]

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

Si implementino in Python le classi **Insegnamento** e **InsegnamentoMutuato**.

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

- **_codice** (tipo **int, Final**)
- **_nome** (tipo **str**)
- **_corso_di_studio** (tipo **str**)

ed i seguenti metodi di istanza:

- *__init__* che inizializza un oggetto della *classe* **Insegnamento** assegnando codice, nome e corso di studio
- proprietà (`@property`) in lettura per tutti gli attributi, con lo stesso nome (senza _);
- proprietà in scrittura per nome e corso di studio.
- metodo **__eq__ (self, o:object)** che restituisce **True** se **o** è un Insegnamento con lo stesso codice di self, (indipendentemente dal nome e dal corso di studio), **False** altrimenti.
- metodo **__str__ (self)** che restituisce la rappresentazione come stringa di un Insegnamento, nella forma `Insegnamento del corso di studio` **`z:`** `codice` **`x`** `e nome` **`y`** dove x, y e z rappresentano il codice, il nome e il corso di studio dell'insegnamento **self**, rispettivamente.

[4 punti] La *classe* **InsegnamentoMutuato** estende Insegnamento ed ha, inoltre, il seguente attributo:

- **_mutua_su** (tipo **Insegnamento**)

La *classe* **InsegnamentoMutuato** ha i seguenti metodi di istanza:

- *__init__* che inizializza un oggetto della *classe* **InsegnamentoMutuato** prendendo in input gli opportuni parametri (incluso l'insegnamento su cui si mutua) e richiamando opportunamente il metodo __init__ della superclasse
- proprietà (`@property`) **mutua_su** in sola lettura per l'attributo **_mutua_su**.

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

- una variabile di istanza **insegnamenti** (list di Insegnamento), contenente tutti gli insegnamenti di tutti i corsi di studio dell'ateneo.

[6 punti] Aggiungere alla classe **Ateneo** i seguenti metodi:

- **`estrai_cds (self) -> list[str]`**
  che restituisce una lista contenente, senza ripetizioni, tutti i corsi di studio aventi almeno un insegnamento nella lista **`self.insegnamenti`**
- **`conta_non_mutui (self, corso_di_studio:str) -> int`**
  che restituisce il numero di insegnamenti **non mutuati** relativi al corso di studio **`corso_di_studio`**.
- **`corsi_con_più_insegnamenti_non_mutuati(self) -> list[str]`**
  che restituisce una lista contenente, senza ripetizioni, tutti i corsi di studio aventi il maggior numero di insegnamenti non mutuati.

Regole per lo svolgimento della prova scritta:

- Per svolgere il compito si hanno a disposizione **100** 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 (smartphone, smart*, calcolatrici, palmari, computer, etc…) deve essere **completamente disattivato** e **depositato in vista sulla cattedra**
- Mettere in vista sul banco un valido documento di identità.

Cognome e Nome _____

Matricola _____

Programmazione e Algoritmi 2 A.A. 2025/2026

Esonero del 17 novembre 2025

Compito n° 2

Esercizio 1 [10 punti]

Si consideri il seguente codice Python.

```
class A:
    def __eq__(self, o: object) -> bool:
        print ("eq di A")
        if isinstance(o, A): return True
        return NotImplemented

class B:
    def __eq__(self, o: object) -> bool:
        print ("eq di B")
        if isinstance(o, B): return True
        if isinstance(o, A): return True
        return NotImplemented

class C (A):
    pass

class D (B):
    pass

class E (C, B):
    pass

class F (B, E):
    pass

c = C()
d = D()
e = E()
f = F()

print ("d==c:")
print (d==c)

print ("c==d:")
print (c==d)
```

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

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

Si scriva nel riquadro a destra cosa **esclusivamente** ciò che viene stampato a video dal codice.

Nel caso in cui il codice dia errore in qualche punto (poiché ad esempio non è possibile calcolare qualche MRO), escludere dal codice il numero minimo di linee in modo da non ottenere errori, commentandole con **#** sul listato stampato in alto su questo foglio, e quindi stampare cosa risulta dal nuovo codice così ottenuto.

Esercizio 2 [10 punti]

Si consideri la seguente sequenza/array di numeri interi:

[1, 50, 7, 25, 9, 30, 15, 73, 8]

3. **[5 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.
4. **[5 punti]** Mostrare passo-passo l'esecuzione della procedura **build-max-heap** sull'array indicato.

Esercizio 3 [14 punti]

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

Si implementino in Python le classi **Insegnamento** e **InsegnamentoMutuato**.

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

- **_codice** (tipo **int, Final**)
- **_nome** (tipo **str**)
- **_corso_di_studio** (tipo **str**)

ed i seguenti metodi di istanza:

- *__init__* che inizializza un oggetto della *classe* **Insegnamento** assegnando codice, nome e corso di studio
- proprietà (`@property`) in lettura per tutti gli attributi, con lo stesso nome (senza _);
- proprietà in scrittura per nome e corso di studio.
- metodo **__eq__ (self, o:object)** che restituisce **True** se **o** è un Insegnamento con lo stesso codice di self, (indipendentemente dal nome e dal corso di studio), **False** altrimenti.
- metodo **__str__ (self)** che restituisce la rappresentazione come stringa di un Insegnamento, nella forma `Insegnamento del corso di studio` **`z:`** `codice` **`x`** `e nome` **`y`** dove x, y e z rappresentano il codice, il nome e il corso di studio dell'insegnamento **self**, rispettivamente.

[4 punti] La *classe* **InsegnamentoMutuato** estende Insegnamento ed ha, inoltre, il seguente attributo:

- **_mutua_su** (tipo **Insegnamento**)

La *classe* **InsegnamentoMutuato** ha i seguenti metodi di istanza:

- *__init__* che inizializza un oggetto della *classe* **InsegnamentoMutuato** prendendo in input gli opportuni parametri (incluso l'insegnamento su cui si mutua) e richiamando opportunamente il metodo __init__ della superclasse
- proprietà (`@property`) **mutua_su** in sola lettura per l'attributo **_mutua_su**.

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

- una variabile di istanza **insegnamenti** (list di Insegnamento), contenente tutti gli insegnamenti di tutti i corsi di studio dell'ateneo.

[6 punti] Aggiungere alla classe **Ateneo** i seguenti metodi:

- **`estrai_cds (self) -> list[str]`**
  che restituisce una lista contenente, senza ripetizioni, tutti i corsi di studio aventi almeno un insegnamento nella lista **`self.insegnamenti`**
- **`conta_non_mutui (self, corso_di_studio:str) -> int`**
  che restituisce il numero di insegnamenti **non mutuati** relativi al corso di studio **`corso_di_studio`**.
- **`corsi_con_più_insegnamenti_non_mutuati(self) -> list[str]`**
  che restituisce una lista contenente, senza ripetizioni, tutti i corsi di studio aventi il maggior numero di insegnamenti non mutuati.

Regole per lo svolgimento della prova scritta:

- Per svolgere il compito si hanno a disposizione **100** 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 (smartphone, smart*, calcolatrici, palmari, computer, etc…) deve essere **completamente disattivato** e **depositato in vista sulla cattedra**
- Mettere in vista sul banco un valido documento di identità.