Cognome e Nome _____

Matricola _____

Programmazione e Algoritmi 2 A.A. 2024/2025

Appello dell'8 luglio 2025

Esercizio 1 [6 punti]

Si consideri il seguente codice Python.

```
class A:
    def __init__ (self, n:int):
        self._n= n

    @property
    def n (self) -> int:
        return self._n + 4

class B:
    _n:int = 0
    @property
    def n (self) -> int:
        return self._n * 3
```

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

class D (C, A):
    pass

class E (B, D):
    pass

c = C(1)
print ("C " + str(c.n))

e = E(2)
print ("E " + str(e.n))
```

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	

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 [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]

Tutto il codice richiesto deve essere in **Python** ed annotato con i tipi opportuni.

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

[4 punti] Si scriva una classe astratta **Movimento** con

- una variabile di istanza **id_sim** (una stringa), che contiene il numero telefonico associato alla SIM a cui il movimento si riferisce;
- una variabile di istanza **tempo** (un intero), che rappresenta l'istante del movimento espresso in secondi trascorsi dalle ore 00:00 del 1° gennaio 2000.

e i seguenti metodi di istanza:

- `__init__` che inizializza un oggetto della classe **Movimento** assegnando **id_sim** e **tempo**;
- Proprietà `id_sim` e `tempo` (in lettura)
- metodo astratto `get_importo(self) -> float` 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** (un intero) che contiene la durata in secondi della chiamata;
- una variabile di istanza **id_sim_destinatario** (una stringa), che contiene l'id_sim del numero chiamato.

La classe **Sms** deve avere:

- una variabile di istanza **id_sim_destinatario** (una stringa), che contiene l'id_sim del destinatario dell'SMS.

La classe **Ricarica** deve avere:

- una variabile di istanza **importo** (un float), 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 tutte le opportune proprietà per accesso in lettura alle variabili di istanza.

[4 punti] Si scriva la sottoclasse Ricarica, incluso il metodo `__init__` (che prende in input gli opportuni parametri e che richiama opportunamente il metodo `__init__` della classe **Movimento**), e proprietà di accesso in lettura per tutti i campi, e metodo `get_importo(self) -> float`.

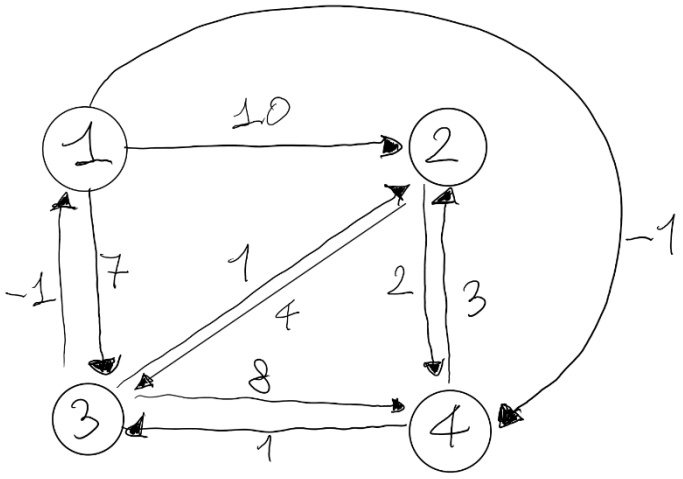
Si assuma, senza scrivere nulla, l'esistenza di una classe **Gestore** con

- una variabile di istanza **movimenti** (list di Movimento), contenente tutti i movimenti di tutte le sim.

[3 punti] Aggiungere alla classe **Gestore** un metodo `get_minuti_tra_due (self, id_sim_1:string, id_sim_2:string) -> int` che restituisce il tempo totale in minuti nel quale le due sim **id_sim_1** e **id_sim_2** 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.