Cognome e Nome _____

Matricola _____

Programmazione e Algoritmi 2 A.A. 2024/2025

Appello del 30 gennaio 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 + 2

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

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

class D (B, A):
    pass

c = C(10)
print (c.n)

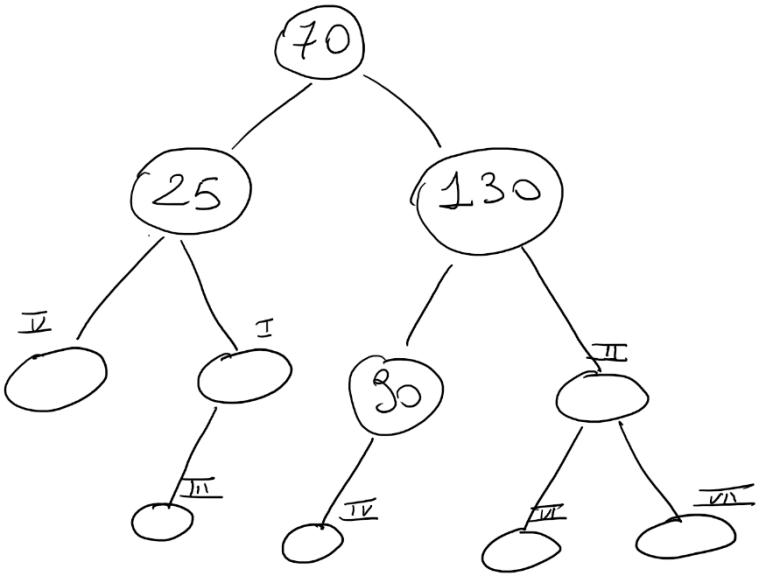
d = D(20)
print (d.n)
```

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	

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

Esercizio 2 [6 punti]

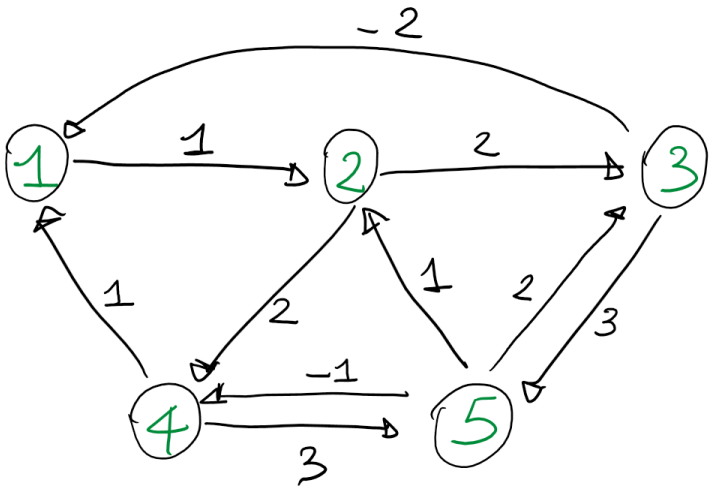


1. **[3 punti, da svolgere su questo foglio]** Si consideri l'albero binario di ricerca in figura e si fornisca una sequenza di 7 valori interi che, inseriti uno dopo l'altro nell'ABR, vadano a riempire i nodi vuoti nell'ordine indicato dai numeri romani accanto a ciascun nodo (si faccia in modo che nell'albero non ci siano valori duplicati):

2. **[3 punti]** Effettuare una **visita in ampiezza** del grafo corrispondente all'ABR del punto precedente, a partire dal nodo radice 70. A partire da un **heap di massimo** vuoto, inserire i valori dell'ABR nell'ordine della visita appena fatta, mostrando l'evoluzione passo passo dell'heap.

Esercizio 3 [11 punti]

Si consideri il grafo (diretto) in figura, per il quale si vogliono calcolare i **cammini minimi tra tutte le coppie di nodi.**



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.

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.

Esercizio 4 [11 punti]

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

Si assuma, senza scrivere nulla, l'esistenza di una classe **Libro** con i seguenti attributi di istanza:

- _codice (un intero)
- _titolo (una stringa)
- _autori (una stringa)

ed i seguenti metodi di istanza:

- __init__ che inizializza un oggetto della classe Libro assegnando codice, titolo e autori
- 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 __hash__ (self) che restituisce l'hash dell'oggetto corrente in modo opportuno
- metodo __str__ (self) che restituisce la rappresentazione come stringa di un libro

[4 punti] Si implementi la classe **Locazione** con i seguenti attributi di istanza:

- _armadio (un intero)
- _scaffale (un intero)
- _posizione (un intero)

ed i seguenti metodi di istanza:

- __init__ che inizializza un oggetto della classe Locazione assegnando armadio, scaffale e posizione;
- 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 oggetto della classe Locazione con stessi armadio, scaffale e posizione di self, False altrimenti.
- metodo __hash__ (self) che restituisce l'hash dell'oggetto corrente in modo opportuno
- metodo __str__ (self) che restituisce la rappresentazione come stringa dell'oggetto corrente, nella forma `(a, s, c)` dove a, s e c rappresentano l'armadio, lo scaffale e la posizione della Locazione self, rispettivamente.

[7 punti] Si implementi la classe **Biblioteca** con i seguenti attributi di istanza (si assuma che non possono essere presenti in biblioteca due libri con lo stesso codice):

- **_locazione** (un **dizionario** con chiavi **Libro** e valori **Locazione** che associa ad in libro la sua locazione in biblioteca)
- **_contenuto** (un **dizionario** con chiavi **Locazione** e valori **Libro** che associa ad una locazione della biblioteca il libro in essa contenuto)

ed i seguenti metodi di istanza:

- *__init__*, che inizializza i due dizionari come dizionari vuoti
- metodo **add_libro (self, libro:Libro, loc: Locazione)** che, se la locazione **loc** è libera, inserisce in biblioteca nella locazione **loc** il libro **libro** (se **libro** è già presente in biblioteca, deve essere spostato dalla attuale locazione, che viene dunque liberata, alla locazione **loc**). Si aggiornino opportunamente entrambi i dizionari.
- metodo **get_locazione (self, libro:Libro) -> Locazione|None** che restituisce la locazione in cui è presente il libro **libro** (**None** se il libro non è in biblioteca).
- metodo **get_libro (self, loc:Locazione) -> Libro|None** che restituisce il libro presente nella locazione **loc** (**None** se la locazione **loc** è libera o inesistente).

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.