# Laboratorio 12 – Esercizi

*Elementi di Informatica e Programmazione*

# Lab 12 – Es 1

Scrivere il modulo **stackandqueue.py** contenente la definizione della classe **Stack** e della classe **Queue** viste a lezione, con la sola modifica delle eccezioni sollevate, che devono essere, rispettivamente, **EmptyStackError** e **EmptyQueueError**.

# Lab 12 – Es 2

Scrivere il programma **checkMatchingBrackets.py** che verifichi la corretta corrispondenza di parentesi chiuse e aperte (tonde, quadre e graffe) all'interno di un testo acquisito tramite lo standard input, interrompendosi al primo errore individuato e fornendo una segnalazione d'errore che sia opportunamente informativa.

Verificarne il corretto funzionamento fornendo in ingresso, mediante redirezione, alcuni sorgenti Python privi di errori e alcuni che contengano errori nelle parentesi.

Per acquisire in input un file inviato mediante redirezione e che sia privo di sentinelle (ad esempio, che NON termini con una riga vuota), si può sfruttare il fatto che la funzione predefinita **input** solleva l'eccezione **EOFError** se viene invocata oltre la fine del file.

# Lab 12 – Es 3

Nel file **memoryCell.py**, progettare un modulo contenente la classe **MemoryCell**, che rappresenti una cella di memoria capace di memorizzare un valore di tipo **int**:

```
class MemoryCell :
  def __init__(self, v) :
    self._val = v

  def getVal(self) :
    return self._val

  def setVal(self, newVal) :
    self._val = newVal

  def clear(self) :
    self.setVal(0)
```

Aggiungere al modulo la classe **backupMemoryCell**, derivata dalla precedente, che sia in grado, quando venga invocato il suo metodo **restore** privo di parametri, di ripristinare il valore immediatamente precedente all'ultima operazione di variazione del valore contenuto al suo interno.

Due invocazioni consecutive di **restore**, senza che venga modificato il contenuto della cella in altro modo, sono da considerarsi una violazione di stato, segnalata sollevando l'eccezione **IllegalStateError**, da progettare.

Aggiungere al modulo opportuno codice di collaudo per entrambe le classi.

# Lab 12 – Es 4

Nel modulo **stackWithSpecialPush.py**, scrivere la classe **Stack** vista a lezione.

Successivamente, nello stesso file, progettare la classe **SpecialStack**, derivata da **Stack**, che le seguenti funzionalità aggiuntive:

- il metodo **contains(element)** restituisce **True** se e solo se la pila contiene un dato uguale al parametro **element** ricevuto; il metodo può accedere direttamente alla variabile di esemplare della superclasse che contiene i dati (senza modificarne il contenuto)
- il metodo **push(element)** deve essere sovrascritto in modo che inserisca effettivamente il dato **element** nella pila se e solo se l'invocazione di **contains** con tale dato restituisce **False (altrimenti, "fallisce silenziosamente")**

In pratica, **SpecialStack** è una pila che non accetta di inserire elementi duplicati.

Aggiungere al modulo opportuno codice di collaudo.

# Lab 12 – Es 5

Scrivere il programma **sortStack.py** contenente la definizione della funzione **sortStackUsingTwoStacks** che riceve una pila di numeri e ne modifica il contenuto in modo che al suo interno i numeri siano ordinati in senso non decrescente, con il numero minimo in cima alla pila.

**La funzione non deve usare liste** né altre strutture dati, ma **soltanto pile**.

Si suggerisce di prendere spunto dall'algoritmo di ordinamento per selezione. Si tenga presente che **sono sufficienti altre due pile**, oltre a quella ricevuta come argomento e che dovrà contenere il risultato dell'elaborazione.

Per collaudare il corretto funzionamento della funzione, rendere eseguibile la classe progettando una funzione **main** che esegua il seguente algoritmo:

- acquisisce dalla riga di comando la dimensione della pila con cui eseguire il collaudo (ovviamente, un numero intero positivo)
- genera una sequenza di numeri casuali in virgola mobile e li inserisce in una pila
- invoca la funzione **sortStackUsingTwoStacks** appena progettata
- estrae dalla pila tutti i numeri e li visualizza sullo standard output, uno per riga, interrompendo il programma con una segnalazione d'errore se i numeri estratti dalla pila non sono in ordine non decrescente

Migliorare, ora, l'algoritmo in modo da svolgere la stessa elaborazione usando **una sola pila aggiuntiva**, progettando la funzione **sortStackUsingOneStack** da invocare nella funzione **main** al posto della funzione precedente.

**MOLTO IMPORTANTE: (come sempre...) individuare un algoritmo che risolva il problema e verificarlo con cura PRIMA di iniziare a scrivere il codice.**