**Esercizio 1**

Si consideri il seguente programma Python:

def f1(LO,LU,LD):

w = len(LO)

k = len(LD)

so = 0

su = 0

sd = 0

for i in range(w):

so += LO[i]

su += LU[i]

if i < k:

sd += LD[i]

return so / w == su / w and su / w == sd / k

def f2(m):

LO = []

LU = []

LD = []

for i in range(len(m)):

for j in range(len(m)):

if i < j:

LO.append(m[i][j])

if i > j:

LU.append(m[i][j])

if i == j:

LD.append(m[i][j])

return f1(LO,LU,LD)

m = [ [6, 2, 6, 4],

[5, 5, 6, 2],

[5, 7, 3, 4],

[1, 2, 4, 2] ]

print(f2(m))

Si descriva sinteticamente la funzione svolta dal programma e si mostrino l'output prodotto e la traccia d'esecuzione.

**Esercizio 2**

Si scriva una funzione *estrai\_lista* che riceve in ingresso due liste di interi *L1* ed *L2* e restituisce una lista contenente alcuni degli elementi di *L1*. In particolare, la lista restituita contiene l'elemento *L1[i] = v* se e solo se il valore *v* compare almeno *v* volte in *L2* in posizioni aventi indice maggiore o uguale ad *i*.

*Esempio*: Se *L1 =* [2,3,4,2,1] ed *L2* = [1,2,3,2,3,1,3,3], allora la funzione restituisce la lista [2,3,1] perché:

* l'elemento *L1[0]* = 2 compare 2 volte in *L2* in posizioni aventi indice maggiore o uguale a 0;
* l'elemento *L1[1]* = 3 compare 4 volte in *L2* in posizioni aventi indice maggiore o uguale a 1;
* l'elemento *L1[2]* = 4 non compare in *L2* in posizioni aventi indice maggiore o uguale a 2;
* l'elemento *L1[3]* = 2 compare 1 volta in *L2* in posizioni aventi indice maggiore o uguale a 3;
* l'elemento *L1[4]* = 1 compare 1 volta in *L2* in posizioni aventi indice maggiore o uguale a 4.

**Esercizio 3**

Si vuole realizzare un’applicazione per la gestione dei dati relativi agli abbonamenti a servizi di streaming di contenuti multimediali. Le informazioni relative ai servizi sono memorizzate in una matrice *S* avente 3 colonne, in cui la generica riga *[n\_servizio, p\_base, d\_creazione]* rappresenta il fatto che il servizio con nome *n\_servizio* ha prezzo base *p\_base* ed è stato creato in data *d\_creazione*. Le informazioni relative agli abbonamenti sono memorizzate in una matrice *A* avente 4 colonne, in cui la generica riga *[n\_utente, n\_servizio, d\_attivazione, tipo]* rappresenta il fatto che l'utente con nome *n\_utente* ha attivato un abbonamento al servizio con nome *n\_servizio* in data *d\_attivazione* e che l'abbonamento è di tipo *tipo* (*'privato'* o *'commerciale'*). Le date sono sempre rappresentate mediante numeri interi. L'*incasso* derivante da un abbonamento è uguale al suo prezzo base se l'abbonamento è di tipo privato – l'incasso è il doppio del prezzo base se l'abbonamento è di tipo commerciale. L'*incasso totale* di un servizio è la somma degli incassi derivanti dagli abbonamenti al servizio stesso.

Si scrivano in Python le seguenti funzioni (la scrittura di eventuali funzioni ausiliarie è ovviamente ammessa):

* *verifica(S,A)*, che restituisce *True* se e solo se si verificano le seguenti condizioni:
  + ogni abbonamento è relativo ad un servizio esistente (cioè il servizio a cui l'abbonamento fa riferimento è presente nella matrice *S*);
  + la data di attivazione di ogni abbonamento è maggiore o uguale alla data di creazione del servizio a cui l'abbonamento fa riferimento;
  + nessun utente ha sottoscritto più abbonamenti ad uno stesso servizio.
* *servizio\_redditizio(S,A)*, che restituisce il nome del servizio avente il massimo incasso totale. Se più di un servizio soddisfa la condizione, la funzione ne restituisce uno qualsiasi.
* *statistiche(A,d1,d2)*, che restituisce una matrice avente 2 colonne, in cui la generica riga [*n\_utente*, *c\_attivazioni*] rappresenta il fatto che l'utente con nome *n\_utente* ha attivato un numero di abbonamenti pari a *c\_attivazioni* nelle date comprese tra *d1* e *d2* (incluse). Se un utente non ha attivato alcun abbonamento nell’intervallo specificato, allora il valore di *c\_attivazioni* è pari a 0.
* *utente\_attivo(A)*, che restituisce l'utente che ha attivato il massimo numero di abbonamenti. Se più di un utente soddisfa la condizione, la funzione ne restituisce uno qualsiasi.

|  |  |  |
| --- | --- | --- |
| 'Servizio A' | 15 | 10 |
| 'Servizio B' | 10 | 30 |
| 'Servizio C' | 12 | 30 |
| 'Servizio D' | 10 | 20 |

|  |  |  |  |
| --- | --- | --- | --- |
| 'Bianchi' | 'Servizio A' | 12 | 'privato' |
| 'Rossi' | 'Servizio B' | 40 | 'commerciale' |
| 'Verdi' | 'Servizio C' | 35 | 'privato' |
| 'Bianchi' | 'Servizio C' | 30 | 'commerciale' |
| 'Bianchi' | 'Servizio D' | 25 | 'privato' |
| 'Neri' | 'Servizio B' | 35 | 'privato' |
| 'Rossi' | 'Servizio A' | 15 | 'commerciale' |

*Esempio*: Se *S =* e *A* = allora:

* *verifica(S,A)* restituisce *True* perché tutte le condizioni sono verificate.
* *servizio\_redditizio(S,A)* restituisce 'Servizio A' perchè l'incasso totale per tale servizio è 45 (gli altri incassi totali sono 30 per 'Servizio B', 36 per 'Servizio C' e 10 per 'Servizio D').

|  |  |
| --- | --- |
| 'Bianchi' | 2 |
| 'Rossi' | 1 |
| ‘Verdi’ | 0 |
| ‘Neri’ | 0 |

* *statistiche(A,15,30)* restituisce la matrice
* *utente\_attivo(A)* restituisce l'utente 'Bianchi' perché tale utente ha attivato 3 abbonamenti ('Rossi' ne ha attivati 2, 'Verdi' 1 e 'Neri' 1).