

Esercizio 1

Data la seguente porzione di programma, rispondere alle domande corrispondenti:

```
int* matricola = new int[6] {i sei numeri che compongono la tua matricola};
```

//1. La seguente istruzione è corretta? Se sì, cosa stampa?

```
cout << *(&matricola[4]);
```

//2. La seguente istruzione è corretta? Se sì, cosa stampa?

```
cout << *(matricola + 3);
```

//3. Cosa stampa la seguente porzione di codice?

```
int& second = matricola[1];
```

```
matricola[1]++;
```

```
second ++;
```

```
cout << second;
```

//4. Come andrebbe deallocata la memoria dinamica allocata inizialmente?

A: delete matricola[6];

B: delete[] matricola;

C: delete matricola;

D: Nel main non va deallocata alcuna memoria dinamica.

Esercizio 2

Implementare una classe Partita che contenga i seguenti campi privati

```
string squadraCasa,squadraOspite
```

```
int goalCasa, goalOspite
```

dove squadraCasa e squadraOspite rappresentano i nomi delle squadre che hanno disputato la partita, mentre goalCasa e goalOspite rappresentano i goal realizzati dalle corrispondenti squadre. Per questa classe, implementare il costruttore con tutti i parametri e i metodi get/set.

Implementare, successivamente, una classe GestorePartite che contenga un solo campo privato

```
list<Partita*> partite
```

ed implementi i seguenti metodi:

- void aggiungiPartita(string squadraCasa, string squadraOspite, int goalCasa, int goalOspite) che aggiunga, dopo averla creata dinamicamente e usando i parametri ricevuti, una partita alla lista *partite*;
- string getSquadraPiuForte() const, che restituisca il nome della squadra che ha vinto più partite, tra tutte quelle (in casa e fuori casa) presenti nella lista *partite*;
- double mediaGoal() const, che restituisca la media dei goal realizzati nelle partite presenti nella lista *partite*;
- costruttore di copia;

Esercizio 3

Scrivere una funzione **esercizio3** che prenda in input un grafo orientato e pesato G , dove ogni nodo in G ha associato un valore intero positivo chiamato deposito. I pesi sugli archi sono interi strettamente maggiori di zero. La funzione deve restituire *true* se per ogni nodo v del grafo valgono le seguenti due condizioni:

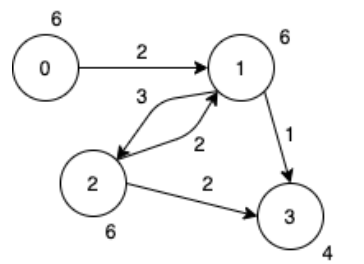
- La somma dei pesi degli archi entranti in v è minore del suo deposito, e
- Il deposito di ogni nodo avente un arco entrante in v è maggiore o uguale al deposito di v .

Se un nodo non ha archi entranti, allora le due condizioni si assumono vere. Se almeno una di tali condizioni non è soddisfatta la funzione deve restituire *false*.

Il grafo è rappresentato da una classe GrafoPesato con la seguente interfaccia (con g un'istanza della classe):

- $g.n()$ restituisce il numero di nodi del grafo,
- $g.m()$ restituisce il numero di archi del grafo,
- $g(i,j)$ restituisce il peso dell'arco tra il nodo i e il nodo j , altrimenti 0 (l'arco non esiste),
- $g.dep(i)$ restituisce il deposito del nodo i .

I nodi sono etichettati da 0 a $g.n()-1$.

| | |
|---|--|
| <p><i>Esempio:</i> in questo caso, la funzione dovrebbe restituire <i>true</i> poiché le condizioni di cui sopra valgono per ogni nodo. Ad esempio:</p> <ul style="list-style-type: none">- il nodo 0 non ha archi entranti, quindi le proprietà si assumono vere;- il nodo 1 ha deposito pari a 6 e due archi entranti, la cui somma è 4, quindi minore del deposito; inoltre, i nodi dei suoi archi entranti hanno tutti il deposito uguale a 6, |  |
|---|--|

Esercizio 4

Scrivere una funzione **esercizio4** che prenda in input un vettore V di elementi (stringhe), un vettore T di transazioni (una transazione è vettore di stringhe v , con v appartenente a V) ed un intero positivo k . La funzione deve restituire *true* se esiste un sottoinsieme W di V tale che le seguenti condizioni sono soddisfatte:

- ogni coppia di elementi v_1 e v_2 in W non appare mai nella stessa transazione, e
- W ha almeno k elementi.

Se non esiste un sottoinsieme W di V tale che queste condizioni siano soddisfatte, la funzione deve restituire *false*.

Si può assumere che:

- V è rappresentato da un `vector<string>`,
- T è rappresentato da un `vector<vector<string>>`,
- Una transazione non contiene mai due stringhe uguali.

| | |
|---|---|
| <p><i>Esempio:</i> una possibile soluzione è data dal sottoinsieme $W = \{a, e\}$. Infatti, nessuna coppia di elementi in W appare insieme in una stessa transazione, e W ha 2 elementi.</p> | <p>$V = [a, b, c, d, e, f, g, h]$ $T = [[a, b, f, c], [b, h, d], [b, a, c], [d, h], [e, f, g, h]]$ $k = 2$</p> |
|---|---|