Esercizio 1

Data la seguente porzione di programma, rispondere alle domande corrispondenti:
int voti[6] = {i sei numeri interi si ottengono dalle sei cifre della matricola};

```
//1. La seguente istruzione è corretta? Se sì, cosa stampa?
        cout << *voti[0];
//2. La seguente istruzione è corretta? Se sì, cosa stampa?
        cout << voti[0] + *(voti+2);
//3. Cosa stampa la seguente porzione di codice? Qual è il significato di tale porzione di codice?
        double s = 0;
        for(int i = 0; i < 6; i++)
                s += *(voti+i);
        if(s/6 > 20)
                cout << “OK”;
        else
                cout <<”NO”;
//4. Cosa stampa la seguente porzione di codice?
        int a = voti[1];
        --a;
        cout << voti[1];
```

Esercizio 2

Implementare una classe Carriera che contenga almeno i seguenti campi privati:
int numAppelli, bool* esiti, string* nomiEsami. In particolare:

- *numAppelli* rappresenta il numero di appelli cui ha partecipato, durante la sua carriera, un dato studente;
- *esiti* è un array di bool tale per cui esiti[i] è True se il tentativo i-esimo è stato positivo (lo studente ha superato il relativo appello) ed è False altrimenti;
- *nomiEsami* è un array di stringhe tale per cui nomiEsami[i] è il nome dell'esame relativo all'i-esimo appello sostenuto dallo studente.

Es: Se uno studente ha partecipato a due appelli di “Analisi” con esito negativo, poi ad un appello di “Fisica” con esito positivo e poi, infine, ad un appello di “Analisi” con esito positivo, si avrà numAppelli = 4, esiti = [False,False,True,True], nomiEsami =[“Analisi”,”Analisi”,”Fisica”,”Analisi”].
Implementare i seguenti metodi:

- Costruttore senza parametri;
- Costruttore di copia;
- Operatore di assegnamento;
- Distruttore;
- void aggiungiTentativo(string nome, bool esito) che, ricevuti come parametri una stringa rappresentante il nome dell'esame sostenuto e un bool rappresentante il relativo esito, li aggiunga, rispettivamente, nei vettori nomiEsami ed esiti;
- void rimuoviUltimoAppello() che rimuove le informazioni (nome ed esito) relative all'ultimo appello inserito (nell'esempio della traccia, l'ultimo tentativo per l'esame di Analisi).
- bool studenteDiligente() che restituisce True se il numero di esiti positivi è maggiore del numero di esiti negativi.

Si specifica che nell'implementare la classe è possibile aggiungere altri campi ritenuti necessari (o fortemente utili).

Esercizio 3

Scrivere una funzione **esercizio3** che prenda in input due interi k_1 e k_2, e un grafo orientato e pesato G dove ogni arco in G ha associato un valore intero chiamato transfer. La funzione deve ritornare la stringa YES se le seguenti condizioni sono soddisfatte:

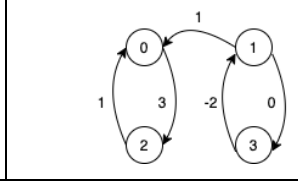
- Il valore di ogni transfer deve essere almeno k_1 e al più k_2;
- Per ogni nodo con grado (n. di archi) di entrata pari, la somma dei transfer entranti in un nodo non può essere maggiore della somma dei transfer uscenti dallo stesso nodo.
- Per ogni nodo con grado (n. di archi) di entrata dispari, la somma dei transfer entranti in un nodo non può essere minore della somma dei transfer uscenti dallo stesso nodo.

Se un nodo non ha alcun arco entrante oppure uscente allora le condizioni si assumono soddisfatte.

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 true se l'arco (i,j) esiste, altrimenti false,
- g.transfer(i,j) restituisce il valore del transfer dell'arco (i,j).

I nodi sono etichettati da 0 a g.n()-1. Se le due condizioni di cui sopra non sono soddisfatte per ogni nodo, allora la funzione restituisce la stringa NO.

Esempio: con k_1 = -3 e k_2 = 3, la funzione deve restituire NO. Infatti, ad esempio, per il nodo 0, che ha grado di entrata pari, la somma dei transfer in entrata, pari a 2 non è maggiore della somma dei transfer in uscita, che è pari a 3. Anche i nodi 2 e 3 soddisfano le condizioni. Tuttavia, il nodo 1 non soddisfa la condizione in quanto il grado di entrata è dispari, ma la somma dei transfer in entrata (-2) è minore della somma dei transfer in uscita (1)	

Esercizio 4

Scrivere una funzione **esercizio4** che prenda in input un vettore S di stringhe, ed un vettore L di liste, dove ogni lista contiene almeno due e al massimo tre stringhe appartenenti a S, e un intero positivo q.
La funzione deve restituire true se esiste un sottoinsieme P di S tale che le seguenti condizioni sono soddisfatte:

- P ha almeno q stringhe,
- Non esiste una lista l in L tale che tutte le sue stringhe sono contenute in P.

Nel caso in cui non esista un sottoinsieme P di S tale che queste condizioni siano soddisfatte, la funzione deve restituire false.

Si può assumere che:

- S è rappresentato da un vector<string> (ArrayList<String> se si usa Java),
- L è rappresentato da un vector<vector<string>> (ArrayList<ArrayList<String>> se si usa Java).

Esempio: in questo esempio, la funzione deve restituire true poiché esiste un sottoinsieme P = [a, e, f] di S tale che le due condizioni sono soddisfatte, ovvero: - P ha almeno q = 3 stringhe, - Non esiste una lista l in L tale che tutte le sue stringhe sono contenute in P	S = [a, b, c, d, e, f,] L = [[a, b, c], [a, d], [d, e], [b, f]] q = 3