Esercizio 2

Consideriamo la classe astratta `StatisticheVettore` che abbia la seguente interfaccia pubblica:

```
class StatisticheVettore {
public:
    virtual double compute(const vector<int>&)   0;
}
```

Mediante l'uso dell'ereditarietà e opportuno overloading del metodo `double compute(const vector<int>&)`, implementare delle classi `MediaVettore`, `ModaVettore` e `MedianaVettore` che restituiscano rispettivamente la *media*, la *moda* e la *mediana* del `vector<int>` parametro:

- La media del vettore $[x_1, ..., x_n]$ è definita come $\frac{1}{n}\sum_{i=0}^{n} x_i$
- La moda del vettore $[x_1, ..., x_n]$ è definita come l'elemento che occorre più volte
- Consideriamo il vettore $x = [x_1, ..., x_n]$. Sia $y = [y_1, ..., y_n]$ il vettore che otteniamo ordinando in senso crescente il vettore X. La mediana di X è definita come y_k se X ha un numero dispari di elementi, altrimenti come $\frac{y_k + y_{k+1}}{2}$ dove $k = \frac{n}{2}$ (divisione intera!).

Realizzare un `main` che mostri l'applicazione del polimorfismo creando opportunamente un `vector` basato solo sulla classe `StatisticheVettore`, ma in cui il primo elemento si comporti come `MediaVettore`, il secondo elemento come `ModaVettore` e il terzo elemento come `MedianaVettore`.

Esercizio 3

Sia G un grafo orientato. Ad ogni nodo v di G è associato un numero intero $W(v)$, detto *peso* del nodo v. Scrivere una funzione che preso in input un grafo orientato G e un vettore di pesi W per i suoi nodi restituisca `true` se e solo se è verificata la seguente proprietà, `false` altrimenti:

Per ogni nodo u di G, la somma dei pesi dei nodi verso i quali esiste un arco uscente da u è maggiore o uguale alla somma dei pesi dei nodi a partire dai quali esiste un arco entrante in u.

La funzione dovrà avere la seguente segnatura:

```
bool esercizio(const Graph& g, const vector<int>& W);
```

Il grafo è rappresentato una classe `Graph` con la seguente interfaccia pubblica:

```
class Graph {
public:
    /* Restituisce il numero di nodi del grafo */
    /* Si può assumere i nodi siano numerati da `0` a `n-1` */
    unsigned n() const;

    /* Restituisce il numero di archi del grafo */
    unsigned m() const;

    /* Restituisce `true` se e solo se esiste un arco da `i` a `j` */
    bool operator()(unsigned i, unsigned j) const;
};
```

Esercizio 4

Scrivere una funzione che presi in input un numero naturale n, una lista di insiemi $(S_1, ..., S_m)$, con $S_i \subseteq \{0, 1, 2, ..., n-1\}$ e un intero k, con $0 < k < m$, determini se esiste un insieme $H \subseteq \{0, 1, 2, ..., n-1\}$ di cardinalità k tale che per ogni insieme S_i nella lista esista almeno un elemento di H contenuto in S_i.