Esercizio n. 1

In una applicazione in cui si richiede di manipolare i dati anagrafici di un gruppo di persone si fa uso delle classi: DatiAnagrafici e VettoreDatiAnagrafici. La prima classe consente di memorizzare le informazioni relative ad una persona, mentre la seconda implementa un array dinamico di oggetti di tipo DatiAnagrafici. Seguono le intestazioni delle classi:

```
class DatiAnagrafici
{
friend ostreamOperator<<(ostream&, const DatiAnagrafici&);
private:
    MyString nome;
    MyString cognome;
    MyString indirizzo;
    MyString codiceFiscale;
    MyString dataDiNascita;
public:
    DatiAnagrafici(MyString n, MyString c, MyString ind, MyString cf,
       MyString dataDiNascita);
    DatiAnagrafici(const DatiAnagrafici&);
    bool operator==(const DatiAnagrafici&);
    MyString getNome();
    MyString getCognome();
    MyString getIndirizzo();
    MyString getCodiceFiscale();
    MyString getDataDiNascita();
    void setNome(MyString);
    void setCognome(MyString);
    void setIndirizzo(MyString);
    void setCodiceFiscale(MyString);
    void setDataDiNascita(MyString);
};
class VettoreDatiAnagrafici
{
friend ostreamOperator<<(ostream&, const VettoreDatiAnagrafici&);
private:
    DatiAnagrafici* dati;
    int lunghezza;
public:
    VettoreDatiAnagrafici(int = 0);
    VettoreDatiAnagrafici(const VettoreDatiAnagrafici&);
    DatiAnagrafici& operator[](int);
    int getLunghezza();
    ~VettoreDatiAnagrafici();
};
```

Si implementino **solo** i metodi di seguito elencati, supponendo già implementati i rimanenti.

- Costruttore della classe DatiAnagrafici
- Metodo getNome() della classe DatiAnagrafici
- Operatore << della classe DatiAnagrafici
- Costruttore di copia della classe VettoreDatiAnagrafici
- Operatore[] della classe VettoreDatiAnagrafici
- Distruttore classe VettoreDatiAnagrafici

Nota: E' possibile aggiungere, qualora lo studente lo ritenga necessario, ulteriori membri pubblici o privati alle classi. Nel caso in cui venga aggiunta una nuova funzione deve esserne fornita l'implementazione.

Esercizio 2

Siano date le classi Studente ed EsameStudente, che rappresentano i dati degli studenti iscritti all'Univerisità, le cui intestazioni sono di seguito riportate:

class Studente
{
public:
  int matricola;
  MyString Nome;
};
	class EsameSostenuto
	{
	public:
	  MyString titolo;
	  int matricola;
	  int voto;
	};

class ListaStudente; class IteratoreStudente;	class ListaEsameSostenuto; class IteratoreEsameSostenuto;

Per ogni studente si memorizzano il nome e la matricola (quest'ultima identifica univocamente uno studente all'interno del sistema), mentre per ogni esame sostenuto da uno studente si memorizza il voto ottenuto.
Si suppone, inoltre, che esistano le classi ListaStudente, IteratoreStudente, ListaEsameSostenuto e IteratoreEsameSostenuto, definite come di consueto, che consentono di rappresentare e manipolare liste di oggetti di tipo Studente ed EsameSostenuto. **Utilizzando gli iteratori** scrivere la funzione:

- ```
  bool studentiConLaStessaMedia(const ListaStudente& lstudenti,
                                const ListaEsameSostenuto& lesami);
  ```

che, data una lista di studenti (*lstudenti*) ed una lista di esami da essi sostenuti (*lesami*), restituisce *true* se tra gli studenti nella lista data ne esistono almeno due aventi la stessa media degli esami sostenuti, *false* altrimenti.
Nota: La media degli esami sostenuti da uno studente si calcola sommando i voti ottenuti (in ogni esame sostenuto) diviso il numero di esami sostenuti; ad esempio, se Gianni ha sostenuto tre esami ottenendo 20, 21 e 22, la sua media è (20+21+22)/3 = 21. Può succedere che due studenti abbiano la stessa media pur avendo sostenuto un numero differente di esami; ad esempio Marco che ha sostenuto due esami ottenendo 20 e 22, ha media (20+22)/2 = 21, che è uguale alla media di Gianni.

Esercizio 3

Data la classe DatiAnagrafici descritta nell'esercizio 1, si utilizzi l'ereditarietà per progettarne una estensione dal nome DatiStudenteUniversitario. La nuova classe deve essere in grado di rappresentare/manipolare, oltre alle informazioni già presenti nella classe DatiAnagrafici, ulteriori informazioni specifiche degli studenti univeristari, quali la matricola (da rappresentare con un intero), il corso di laurea (da rappresentare con MyString) e l'anno di corso (da rappresentare con un intero).
È richiesta l'implementazione dell'interfaccia della classe DatiStudenteUniversitario e di un costruttore.
  ```