Corso Programmazione ad Oggetti
Corso di Laurea in Informatica
Esercitazione di Laboratorio del 22/10/2018

ESERCIZIO 1 Realizzare un main che letti da input alcuni numeri li inserisca in una lista STL (http://www.cplusplus.com/reference/list/list/). Eseguire quindi le seguenti operazioni:

1. Stampare il contenuto della lista.
2. Stampare la dimensione della lista.
3. Stampare il primo intero nella lista.
4. Stampare l'ultimo intero nella lista.
5. Stampare il penultimo elemento presente nella lista .
6. Rimuovere gli interi duplicati e consecutivi nella lista (usare il metodo ***unique***). Stampare nuovamente il contenuto della lista e verificare i duplicati consecutivi siano stati eliminati.
7. Ordinare la lista in modo che gli interi contenuti in essa siano ordinati in ordine crescente (usare il metodo ***sort***). Stampare nuovamente il contenuto della lista e verificare che l'ordinamento sia rispettato.
8. Rimuovere tutti gli interi duplicati nella lista (usando nuovamente ***unique***). Stampare nuovamente il contenuto della lista e verificare tutti duplicati siano stati eliminati.

Ad esempio se la lista contenesse: 1 - 2 - 2 - 7 - 8 - 1 - 1 - 3 al punto 6 si otterrebbe: 1 - 2 - 7 - 8 - 1 - 3 mentre al punto 7: 1 - 1 - 2 - 3 -7 - 8 ed al punto 8: 1 - 2 - 3 - 7 - 8.

ESERCIZIO 2

Utilizzare la classe ***Contatto***, riportata in seguito, per realizzare una classe ***GestoreContatti***.

Dotare la classe di un dato private che sia una lista di contatti, e dei seguenti metodi che permettano di gestire la lista:

- **void stampaContatti ()**, che stampi su standard output il contenuto della lista.
- **bool inserisciContatto (const Contatto& c)**, che aggiunga il contatto c, alla fine della lista, se non esiste già un contatto uguale. Restituire true se è stato inserito, false altrimenti.
- **int numeroContatti ()**, che restituisca il numero di contatti presenti nella lista.
- **void ordinaContatti ()**, che ordini la lista di contatti.
- **bool eliminaContattoConRemove (const Contatto& c)**, che elimini dalla lista il contatto c, se presente. Utilizzare a questo scopo il metodo *remove* di list (http://www.cplusplus.com/reference/list/list/remove/). Restituire true se è stato eliminato, false altrimenti.
- **bool eliminaContattoConErase (const Contatto& c)**, che come il metodo precedente, elimini dalla lista il contatto c, se presente. Tuttavia, in questo caso, si utilizzi il metodo *erase* di list. Si faccia attenzione al fatto che il metodo *erase* invalida gli iteratori che fanno riferimento agli elementi rimossi (http://www.cplusplus.com/reference/list/list/erase/). Restituire true se è stato eliminato, false altrimenti.
- **vector<string> trovaNumeri (const string& nome, const string& cognome)**, che restituisca un vettore contenente i numeri di telefono memorizzati nella lista in corrispondenza dei contatti con il nome ed il cognome ricevuti come parametri.
- **string trovaCognomePiuFrequente ()**, che restituisca una stringa corrispondente al cognome con il maggior numero di occorrenze nella lista. Se la lista è vuota restituire una stringa vuota.
- **bool verificaDueContattiStessoTelefono ()**, che restituisca true se nella lista sono presenti almeno due contatti con nome e/o cognome diversi, ma stesso numero di telefono, false altrimenti.

Corso Programmazione ad Oggetti
Corso di Laurea in Informatica
Esercitazione di Laboratorio del 22/10/2018

Realizzare un main in cui viene creato un oggetto di tipo ***GestoreContatti***, e viene controllata la correttezza dei metodi implementati. Verificare, inoltre, che l'effetto dei due metodi per l'eliminazione sia lo stesso.

File Contatto.h

```
#ifndef CONTATTO_H
#define CONTATTO_H

#include <iostream>
#include <string>
using namespace std;

class Contatto
{
friend istream& operator >> (istream& in, Contatto&);
friend ostream& operator << (ostream& out, const Contatto&);

public:
      Contatto();
      Contatto(const string & nom, const string & cog);
      Contatto(const string & nom, const string & cog, const string& n, const
string& e);

      void setNome(const string & nom);
      void setCognome(const string & cog);
      void setTelefono(const string& telefono);
      void setEmail(const string & email);

      const string& getCognome() const;
      const string& getEmail() const;
      const string& getNome() const;
      const string& getTelefono() const;

      bool operator==(const Contatto& c) const;
      bool operator!=(const Contatto& c) const;
      bool operator<(const Contatto& c) const;
      bool operator>(const Contatto& c) const;

private:
      string nome;
      string cognome;
      string telefono;
      string email;
};

#endif
```

File Contatto.cpp

```
#include "Contatto.h"

Contatto::Contatto() {}

Contatto::Contatto(const string & nom, const string & cog)   : nome(nom),
cognome(cog) {}

Contatto::Contatto(const string & nom, const string & cog, const string& tel,
const string& em)  : nome(nom), cognome(cog), telefono(tel), email(em) {}

ostream& operator << (ostream& out, const Contatto & c)
{
```

```
	out << c.nome << " - ";
	out << c.cognome << " - ";
	out << c.telefono << " - ";
	out << c.email << endl;
	return out;
}

istream& operator >> (istream& in, Contatto & c)
{
	cout<<"Inserire il nome"<<endl;
	in >> c.nome;
	cout<<"Inserire il cognome"<<endl;
	in >> c.cognome;
	cout<<"Inserire l'indirizzo email"<<endl;
	in >> c.email;
	cout<<"Inserire il numero di telefono"<<endl;
	in >> c.telefono;
	return in;
}

bool Contatto::operator==(const Contatto& c) const{
	return	this->nome==c.nome	&&	this->cognome==c.cognome	&&	this-
>email==c.email && this->telefono==c.telefono;
}

bool Contatto::operator!=(const Contatto& c) const{
	return	this->nome!=c.nome	||	this->cognome!=c.cognome	||	this-
>email!=c.email || this->telefono!=c.telefono;
//	Oppure return !(*this==c);
}

bool Contatto::operator<(const Contatto& c) const{
	if(cognome<c.cognome)
		return true;
	if(nome<c.nome)
		return true;
	return false;
}

bool Contatto::operator>(const Contatto& c) const{
	if(cognome>c.cognome)
		return true;
	if(nome>c.nome)
		return true;
	return false;
//	Oppure return (*this)!=c && !((*this)<c);
}

const string& Contatto::getCognome() const {
	return cognome;
}

const string& Contatto::getEmail() const {
	return email;
}

const string& Contatto::getNome() const {
	return nome;
}
```

Corso Programmazione ad Oggetti
Corso di Laurea in Informatica
Esercitazione di Laboratorio del 22/10/2018

```
const string& Contatto::getTelefono() const {
      return telefono;
}

void Contatto::setNome(const string & nom){
      nome = nom;
}

void Contatto::setCognome(const string & cog){
      cognome = cog;
}

void Contatto::setEmail(const string & email){
      this->email = email;
}

void Contatto::setTelefono(const string& telefono) {
      this->telefono = telefono;
}
```

ESERCIZIO 3

Realizzare un programma che legga in input un'espressione matematica (sotto forma di stringa) contenente parentesi (solo tonde) e determini utilizzando la classe Stack template riportata in seguito, se l'espressione è ben parentesizzata (ovvero, ogni parentesi aperta ha una corrispondente parentesi chiusa).

Classe **Stack**:

```
#ifndef STACK_H_
#define STACK_H_

#include <vector>
using namespace std;

template<class T>
class Stack
{
private:
    vector<T> s;
public:
    Stack(){}
    void push(const T& e){ s.push_back(e);}
    bool pop(){ if(s.empty()) return false; s.pop_back(); return true;}
    T top() {return s.back(); }
    bool stackVuoto() {return s.empty();}
    void svuota() { s.clear();}
};

#endif
```

Alcuni suggerimenti per lo svolgimento:

- Non utilizzare caratteri accentati per nomi di campi, variabili o metodi.
- Per il carattere tilde (~) occorre digitare: ALT + 126 su sistemi Windows, ALT gr + ì su sistemi Linux, ALT + 5 su sistemi Mac OS X.
- Per ogni esercizio realizzare una cartella contenente tutti i file creati.

Corso Programmazione ad Oggetti
Corso di Laurea in Informatica
Esercitazione di Laboratorio del 22/10/2018

- Si consiglia di utilizzare un semplice editor di testo (ad esempio, gedit), e compilare da linea di comando. Per compilare occorre passare tutti i file sorgente (.cpp) al compilatore.
- Minimizzare le inclusioni.