Laurea in Informatica – Programmazione ad Oggetti – Appello d'Esame 14/2/2018

Nome..... Cognome..... Matricola.....

Esercizio 1

Si assumano le seguenti specifiche riguardanti la libreria Qt (**attenzione:** non si tratta di codice da definire!).

- `QWidget` è la classe base di tutte le classi Gui della libreria Qt. La classe `QWidget` ha il distruttore virtuale. La classe `QWidget` rende disponibile un metodo `QSize size() const` con il seguente comportamento: `w.size()` ritorna un oggetto di tipo `QSize` che rappresenta la dimensione del widget `w`. Inoltre, la classe `QWidget` rende disponibile un metodo `void resize(const QSize&)` con il seguente comportamento: `w.resize(qs)` imposta la dimensione del widget `w` a `qs`. Qt rende disponibili gli operatori esterni di uguaglianza `bool operator==(const QSize&, const QSize&)` e disuguaglianza `bool operator!=(const QSize&, const QSize&)` tra oggetti di `QSize`.
- `QAbstractButton` è una classe astratta che deriva direttamente e pubblicamente da `QWidget` ed è la classe base astratta di tutti i pulsanti astratti (button widgets). La classe `QAbstractButton` rende disponibile un metodo `bool isDown() const` con il seguente comportamento: `ab.isDown()` ritorna `true` se il button `ab` è nello stato "down", altrimenti ritorna `false`.
- `QCheckBox` è una classe concreta che deriva direttamente e pubblicamente da `QAbstractButton` e rappresenta un pulsante checkbox.
- In Qt esistono altre classi concrete che derivano direttamente e pubblicamente da `QAbstractButton` (ad esempio `QRadioButton`).
- La classe `QAbstractSlider` è una classe astratta che deriva direttamente e pubblicamente da `QWidget` ed è la classe base astratta di tutti i cursori astratti (slider widgets). La classe `QAbstractSlider` rende disponibile un metodo `void setSliderDown(bool x)` che permette di impostare lo stato "down" dello slider quando è invocato con parametro attuale `true`.
- `QSlider` è una classe concreta che deriva direttamente e pubblicamente da `QAbstractSlider`. `QSlider` è dotato di un costruttore di default.
- In Qt esistono altre classi concrete che derivano direttamente e pubblicamente da `QAbstractSlider` (ad esempio `QScrollBar`).

Si assuma una situazione in cui non vi è alcuna condivisione di memoria. Definire una funzione:

```
list<QCheckBox> Fun(vector<const QWidget*>&, const QSize&)
```

con il seguente comportamento: in ogni invocazione `Fun(vec,sz)`, per ogni puntatore `p` appartenente al vector `vec`:

1. se `*p` è un qualsiasi cursore astratto (slider widget) allora:
   - se `*p` non è un `QSlider` allora viene sostituito da uno `QSlider` di default con dimensione impostata a `sz`;
   - se `*p` è invece un `QSlider` di dimensione diversa da `sz` allora la sua dimensione viene impostata a `sz`; inoltre `*p` viene impostato allo stato "down"
2. se invece `*p` non è un cursore astratto (slider widget) ma è un qualsiasi pulsante astratto (button widget) in stato "down", allora il puntatore `p` viene rimosso dal vector `vec`;
3. `Fun(vec,sz)` deve ritornare una lista di `QCheckBox` contenente una copia di tutti gli oggetti `QCheckBox` puntati da un puntatore rimosso dal vector `vec` nel precedente punto 2.

Esercizio 2

```
class A {
  bool x;
public:
  virtual ~A() = default;
};

class B {
  bool y;
public:
  virtual void f() const { cout << "B::f "; }
};

class C: public A {};

class D: public B {
public:
  void f() const { cout << "D::f "; }
};

class E: public D {
public:
  void f() const { cout << "E::f "; }
};

template<class T>
void Fun(const T& ref) {
  try{ throw ref; }
  catch(const C& c) {cout << "C ";}
  catch(const E& e) {cout << "E "; e.f();}
  catch(const B& b) {cout << "B "; b.f();}
  catch(const A& a) {cout << "A ";}
  catch(const D& d) {cout << "D ";}
  catch(...)        {cout << "GEN ";}
}

C c; D d; E e; A& a1 = c; B& b1 = d; B& b2 = e; D& d1 = e; D* pd = dynamic_cast<E*>(&b2);
```

Le precedenti definizioni compilano senza provocare errori (con gli opportuni `#include` e `using`). Per ognuna delle seguenti istruzioni di invocazione della funzione `Fun` scrivere nell'apposito spazio:

- **NON COMPILA** se la compilazione dell'istruzione provoca un errore;
- **ERRORE RUN-TIME** se l'istruzione compila correttamente ma la sua esecuzione provoca un errore a run-time;
- se l'istruzione compila correttamente e non provoca errori a run-time allora si scriva la stampa che l'esecuzione produce in output su `cout`; se non provoca alcuna stampa allora si scriva **NESSUNA STAMPA**.

`Fun(c);`	
`Fun(d);`	
`Fun(e);`	
`Fun(a1);`	
`Fun(b1);`	
`Fun(d1);`	
`Fun(*pd);`	
`Fun<D>(*pd);`	
`Fun<D>(e);`	
`Fun<E>(*pd);`	
`Fun<E>(e);`	
`Fun<E>(d1);`	
`Fun<A>(c);`	

Esercizio 3

```
interface X { public void f(); }

interface Y { public char g(); }

interface Z extends X { public void h(); }

class A implements Y, Z {
    public void f() { System.out.print("A.f() "); }
    public char g() { System.out.print("A.g() "); return 'A'; }
    public void h() { System.out.print("A.h() "); }
    public void m() { System.out.print("A.m() "); }
}

class B implements X {
    public void f() { System.out.print("B.f() "); }
}

class C extends B implements Y {
    public void f() { System.out.print("C.f() "); }
    public char g() { System.out.print("C.g() "); return 'C'; }
}

class D extends A {
    public char g() { System.out.print("D.g() "); return 'D'; }
    public void h() { System.out.print("D.h() "); }
}
```

Le precedenti definizioni compilano senza provocare errori. Si supponga che ognuno dei seguenti frammenti sia il codice di un metodo `main()` di qualche classe che può accedere alle precedenti definizioni. Si scriva nell'apposito spazio contiguo:

- **NON COMPILA** se la compilazione del `main()` provoca un errore;
- **ECCEZIONE** se il `main()` compila correttamente ma la sua esecuzione provoca una eccezione (di qualsiasi tipo);
- se il `main()` compila correttamente e la sua esecuzione non provoca eccezioni allora si scriva la stampa che l'esecuzione produce in output su `System.out`; se non provoca alcuna stampa allora si scriva **NESSUNA STAMPA**.

```
X x = new Z(); x.f(); _____

Z z  = new A(); z.f(); z.m(); _____

Y y  = new C(); X x  = (B)y; x.f(); y.g(); _____

Y y  = new A(); A a  = new D();
a.h(); X x = (D)a; a.g(); y = a; y.g(); _____

Z z = new A(); Y y  = (D)z; y.g(); _____

B b = new C(); b.f(); A a = (A)b; a.f(); _____

Z z  = new D(); ((A)z).h(); Y y = z; y.g(); _____

Y y = new D(); ((Z)y).h(); X x = (A)y; x.f(); _____

X x  = new B(); Y y = new C();
C c1 = (C)x; C c2 = (C)y; c1.g(); c2.g(); _____

Y y = new D(); ((A)y).m(); ((D)y).m(); _____
```