UNIVERSITÀ DELLA CALABRIA
DIPARTIMENTO DI MATEMATICA E INFORMATICA

Corso di Laurea In Informatica
Esame di Programmazione ad Oggetti
29/06/2017
Tempo a Disposizione: 90 min

Cognome: _____ Nome: _____

Matricola: _____ Esercizi lab: _____

```
 1: class Biglietteria{
 2: public:
 3:   Biglietteria():bigliettiVenduti(0),numeroBigliettiVenduti(0),capacitaBigliettiVenduti(0) {};
 4:   Biglietteria(const Biglietteria& d);
 5:   Biglietteria& operator=(const  Biglietteria& d);
 6:   ~Biglietteria();
 7:   void aggiungiBigliettoDaVendere(Biglietto* b);
 8:   void aggiungiBigliettoVenduto(Biglietto* b);
 9:   friend ostream& operator<<(ostream& out, const Biglietteria& d);
 10:  unsigned getNumBigliettiVenduti() const;
 11:  void svuotaBigliettiDaVendere();
 12:  void svuotaBigliettiVenduti();
 13: protected:
 14:  vector<Biglietto*> bigliettiDaVendere;
 15:  Biglietto** bigliettiVenduti;
 16:  unsigned numeroBigliettiVenduti;
 17:  unsigned capacitaBigliettiVenduti;
};

//Completare opportunamente il main (max 2 punti)
int main() {
    // Un biglietto ha un codice identificativo, un prezzo e una stringa rappresentante l'evento
    // a cui si riferisce. Vi sono diverse tipologie di biglietto, ad esempio biglietti per posti
    // in platea o in poltrona numerata. Alcuni di questi hanno dei dati aggiuntivi, ad esempio un
    // biglietto in poltrona numerata ha anche il numero del posto assegnato.
    Biglietto* b1 = new BigliettoPlatea(1,30,"Musical");
    Biglietto* b2 = new BigliettoPoltronaNumerata(2,40,"Concerto",134);

    Biglietteria* biglietteria1 = new Biglietteria();
    biglietteria1->aggiungiBigliettoDaVendere(b1);

    Biglietteria* biglietteria2=biglietteria1;
    biglietteria2->aggiungiBigliettoDaVendere(b2);
    biglietteria2->svuotaBigliettiDaVendere();




    return 0;
}
//Implementare i seguenti metodi (max 10 punti)
Biglietteria&  Biglietteria::operator=(const  Biglietteria& b){









}
```

Dipartimento di Matematica e Informatica
Università della Calabria
Via P. Bucci, Cubo 30B
87036 Rende (CS) - ITALIA

Tel. (+39) 0984.496402
Fax (+39) 0984.496410
dipartimento.demacs@unical.it
dipartimento.demacs@pec.unical.it

www.mat.unical.it

```
/* Aggiunge un biglietto ai biglietti venduti. Il biglietto va aggiunto solo se è presente un
biglietto uguale (ovvero con stesso codice, stesso prezzo e stesso evento) nei biglietti da
vendere. In tal caso, il biglietto viene aggiunto nei biglietti venduti e rimosso dai biglietti da
vendere. Inoltre, i biglietti venduti devono essere inseriti in ordine crescente per codice
identificativo. */
void Biglietteria::aggiungiBigliettoVenduto(Biglietto* b){









}
// Rimuove tutti i biglietti venduti
void Biglietteria::svuotaBigliettiVenduti(){




}
```

```
//Definire opportunamente le seguenti classi (max 3 punti):
class Biglietto {









};
class BigliettoPoltronaNumerata : ____________ Biglietto {









};
```

Dipartimento di Matematica e Informatica
Università della Calabria
Via P. Bucci, Cubo 30B
87036 Rende (CS) - ITALIA

Tel. (+39) 0984.496402
Fax (+39) 0984.496410
dipartimento.demacs@unical.it
dipartimento.demacs@pec.unical.it

www.mat.unical.it

Programmazione Ad Oggetti. 29 Giugno 2017

Cognome: _____ Nome: _____

Matricola: _____ Esercizi lab: _____

Rispondere alle seguenti domande a risposta multipla (2.5 punti per risposta e motivazione esatta, -2 punti per risposta sbagliata, 0 per risposta non data o risposta esatta ma priva di motivazione):

1. Quale tra le seguenti implementazioni è corretta?

   a)
   ```
   Biglietteria::~Biglietteria(){
       delete [] bigliettiVenduti;
   }
   ```

   b)
   ```
   Biglietteria::~Biglietteria(){}
   ```

   c)
   ```
   Biglietteria::~Biglietteria() {
       for(unsigned i=0;i<numeroBigliettiVenduti;++i)
           delete bigliettiVenduti[i];
       delete [] bigliettiVenduti;
       delete bigliettiDaVendere;
   }
   ```

   d)
   ```
   Biglietteria::~Biglietteria() {
       for(unsigned i=0;i<numeroBigliettiVenduti;++i)
           delete bigliettiVenduti[i];
       delete [] bigliettiVenduti;
       for(vector<Biglietto*>::iterator it=bigliettiDaVendere.begin();
           it!=bigliettiDaVendere.end();++it)
               delete *it;
   }
   ```

   Motivazione:

2. Quale tra le seguenti implementazioni è corretta?

   a)
   ```
   void Biglietteria::svuotaBigliettiDaVendere() {
       delete [] bigliettiDaVendere;
   }
   ```

   b)
   ```
   void Biglietteria::svuotaBigliettiDaVendere() {
       bigliettiDaVendere.clear();
   }
   ```

   c)
   ```
   void Biglietteria::svuotaBigliettiDaVendere() {
       for(vector<Biglietto*>::iterator it=bigliettiDaVendere.begin();
           it!=bigliettiDaVendere.end();++it)
               bigliettiDaVendere.erase(it);
   }
   ```

   d)
   ```
   void Biglietteria::svuotaBigliettiDaVendere() {
       for(vector<Biglietto*>::iterator it=bigliettiDaVendere.begin();
           it!=bigliettiDaVendere.end();++it)
               delete *it;
   }
   ```

   Motivazione:

3. Quale tra le seguenti implementazioni è corretta?

   a)
   ```
   ostream& operator<<(ostream& out, const Biglietteria& b){
       for(vector<Biglietto*>::const_iterator it=b.bigliettiDaVendere.begin();
           it!=b.bigliettiDaVendere.end();it++)
               out<<(*it).info();
       return out;
   }
   ```

b)
```
ostream& operator<<(ostream& out, const Biglietteria& b){
    for(vector<Biglietto*>::const_iterator it=b.bigliettiDaVendere.begin();
        it!=b.bigliettiDaVendere.end();it++)
            out<<(*it)->info();
    return out;
}
```

c)
```
ostream& operator<<(ostream& out, const Biglietteria& b){
    out<<b.bigliettiDaVendere->info();
    return out;
}
```

d)
```
ostream& operator<<(ostream& out, const Biglietteria& b){
    out<<"Num biglietti venduti: "<<b.bigliettiVenduti.size()<<endl;
    return out;
}
```

Motivazione:

4. Quale tra le seguenti implementazioni è corretta?

a)
```
bool Biglietto::operator==(const Biglietto& b1, const Biglietto& b2) {
   return b1==b2;
}
```

b)
```
bool Biglietto::operator==(const Biglietto& b1, const Biglietto& b2) {
   return b1->getCodice()==b2->getCodice();
}
```

c)
```
bool Biglietto::operator==(const Biglietto& b) {
   return b.getPrezzo()==this.getPrezzo();
}
```

d) `Nessuna delle precedenti, fornire implementazione:`

Motivazione:

5. Sia **class BiglietteriaConcerto: protected Biglietteria** la definizione di una classe BiglietteriaConcerto. Quale tra le seguenti istruzioni **è consentita** nel costruttore di copia di BiglietteriaConcerto?

a) `cout<<getNumBigliettiVenduti()<<endl;`

b) `bigliettiVenduti = new Biglietto*[10];`

c) `Sono entrambe consentite`

d) `Sono entrambe sbagliate`

Motivazione:

6. Sia **class BiglietteriaConcerto: protected Biglietteria** la definizione di una classe BiglietteriaConcerto. Quale tra le seguenti istruzioni **è consentita** nel main?

a) `BiglietteriaConcerto* b; b->svuotaBigliettiVenduti();`

b) `Biglietteria* b = new BiglietteriaConcerto();`

c) `Sono entrambe consentite`

d) `Sono entrambe sbagliate`

Motivazione: