UNIVERSITÀ DELLA CALABRIA
DIPARTIMENTO DI MATEMATICA E INFORMATICA

Corso di Laurea In Informatica
Esame di Programmazione ad Oggetti
24/07/2017
Tempo a Disposizione: 90 min

Cognome: _____ Nome: _____

Matricola: _____ Esercizi lab: _____

```
 1: class Ristorante {
 2: public:
 3:   Ristorante():ordinazioniServite(0),numeroOrdinazioniServite(0),
                  capacitaOrdinazioniServite(0){};
 4:   Ristorante(const Ristorante& r);
 5:   Ristorante& operator=(const Ristorante& r);
 6:   ~Ristorante();
 7:   bool aggiungiOrdinazioneDaServire(Ordinazione* o);
 8:   bool aggiungiOrdinazioneServita(Ordinazione* o);
 9:   friend ostream& operator<<(ostream& out, const Ristorante& r);
 10:  unsigned getNumeroOrdinazioniServite() const;
 11:  void svuotaOrdinazioniDaServire();
 12:  void svuotaOrdinazioniServite();
 13: protected:
 14:  list<Ordinazione*> ordinazioniDaServire;
 15:  Ordinazione** ordinazioniServite;
 16:  unsigned numeroOrdinazioniServite;
 17:  unsigned capacitaOrdinazioniServite;
};

//Completare opportunamente il main (max 2 punti)
int main() {
    // Un'ordinazione ha un codice identificativo, un prezzo e una descrizione.
    // Inoltre, possono esserci diversi tipi di ordinazioni: "take away" (da portare via), oppure
    // da consegnare a domicilio, oppure ordinazioni classiche ovvero
    // da consumare presso il ristorante.
    // Un'ordinazione da consegnare a domicilio contiene anche il domicilio a cui recapitarla.
    Ordinazione* b1 = new OrdinazioneTakeAway(1,4,"Pizza Margherita");
    Ordinazione* b2 = new OrdinazioneDomicilio(2,15,"Hambuger","Via Rossi 11, Roma");

    Ristorante* ristorante1 = new Ristorante();
    ristorante1->aggiungiOrdinazioneDaServire(b1);
    ristorante1->aggiungiOrdinazioneDaServire(b2);

    Ristorante* ristorante2 = new Ristorante(*ristorante1);
    ristorante2->aggiungiOrdinazioneDaServire(b1);



    return 0;
}
//Implementare i seguenti metodi (max 10 punti)
// Rimuove tutte le ordinazioni servite
void Ristorante::svuotaOrdinazioniServite (){









}
```

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'ordinazione alle ordinazioni da servire. L'ordinazione va aggiunta in coda e
soltanto se non esiste un'altra ordinazione con lo stesso codice identificativo (tra le
ordinazioni da servire). Se l'aggiunta viene effettuata restituire true, altrimenti false. */
bool Ristorante::aggiungiOrdinazioneDaServire(Ordinazione* o);









}
/* Aggiunge un'ordinazione alle ordinazioni servite. L'ordinazione va aggiunta soltanto se esiste
un'ordinazione con lo stesso codice identificativo tra le ordinazioni da servire: in tal caso
l'ordinazione viene rimossa dalle ordinazioni da servire e aggiunta a quelle servite. Se
l'aggiunta viene effettuata restituire true, altrimenti false. */
bool Ristorante::aggiungiOrdinazioneServita(Ordinazione* o){









}_____________________________________________________________________________
//Definire opportunamente le seguenti classi (max 3 punti):
class Ordinazione {









};
class OrdinazioneDomicilio : ___________ Ordinazione {









};
```

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. 24 Luglio 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)
   ```
   Ristorante::~Ristorante(){}
   ```

   b)
   ```
   Ristorante::~Ristorante(){
       delete [] ordinazioniServite;
       delete [] ordinazioniDaServire;
   }
   ```

   c)
   ```
   Ristorante::~Ristorante() {
       for(unsigned i=0;i<numeroOrdinazioniServite;++i)
           delete ordinazioniServite[i];
       delete [] ordinazioniServite;
       delete [] ordinazioniDaServire;
   }
   ```

   d)
   ```
   Ristorante::~Ristorante() {
       delete [] ordinazioniServite;
   }
   ```

   Motivazione:

2. Quale tra le seguenti implementazioni è corretta?

   a)
   ```
   void Ristorante::svuotaOrdinazioniDaServire() {
       ordinazioniDaServire = 0;
   }
   ```

   b)
   ```
   void Ristorante::svuotaOrdinazioniDaServire() {
       ordinazioniDaServire.clear();
   }
   ```

   c)
   ```
   void Ristorante::svuotaOrdinazioniDaServire() {
       for(list<Ordinazione*>::iterator it=ordinazioniDaServire.begin();
           it!=ordinazioniDaServire.end();++it){
               ordinazioniDaServire.erase(it);
               return;
           }
   }
   ```

   d)
   ```
   void Ristorante::svuotaOrdinazioniDaServire() {
       for(list<Ordinazione*>::iterator it=ordinazioniDaServire.begin();
           it!=ordinazioniDaServire.end();++it){
               ordinazioniDaServire.remove(it);
           }
   }
   ```

   Motivazione:

3. Quale tra le seguenti implementazioni è corretta?

   a)
   ```
   ostream& operator<<(ostream& out, const Ristorante& b){
       for(unsigned i=0;i<ordinazioniDaServire.size();i++)
           out<<ordinazioniDaServire[i]->info();
       return out;
   }
   ```

   b)
   ```
   ostream& operator<<(ostream& out, const Ristorante& b){
       out<<b.ordinazioniDaServire.info();
       out<<b.ordinazioniServite.info();
       return out;
   }
   ```

c)
```
ostream& operator<<(ostream& out, const Ristorante& b){
    for(list<Ordinazione*>::const_iterator it=b.ordinazioniDaServire.begin();
        it!=b.ordinazioniDaServire.end();it++)
            out<<(*it)->info();
    return out;
}
```

d)
```
ostream& operator<<(ostream& out, const Ristorante& b){
    for(list<Ordinazione*>::const_iterator it=b.ordinazioniDaServire.begin();
        it!=b.ordinazioniDaServire.end();it++)
            out<<it;
    return out;
}
```

Motivazione:

4. Quale tra le seguenti implementazioni è corretta?

a)
```
bool Ordinazione::operator==(const Ordinazione& o1, const Ordinazione& o2) {
   return *o1==*o2;
}
```

b)
```
bool Ordinazione::operator==(const Ordinazione& o1, const Ordinazione& o2) {
   return o1.getCodice()==o2.getCodice();
}
```

c)
```
bool Ordinazione::operator==(const Ordinazione& o) {
   return o==this;
}
```

d) `Nessuna delle precedenti, fornire implementazione:`

Motivazione:

5. Sia **class RistoranteFastFood: private Ristorante** la definizione di una classe RistoranteFastFood. Quale tra le seguenti istruzioni **è consentita** nel costruttore di default di RistoranteFastFood?

a) `svuotaOrdinazioniDaServire();`

b) `ordinazioniServite = 0;`

c) `Sono entrambe consentite`

d) `Sono entrambe sbagliate`

Motivazione:

6. Sia **class RistoranteFastFood: private Ristorante** la definizione di una classe RistoranteFastFood. Quale tra le seguenti istruzioni **è consentita** nel main?

a) `Ristorante* r = new Ristorante(); r->ordinazioniServite = 0;`

b) `RistoranteFastFood rf; rf.svuotaOrdinazioniDaServire();`

c) `Sono entrambe consentite`

d) `Sono entrambe sbagliate`

Motivazione: