UNIVERSITÀ DELLA CALABRIA
DIPARTIMENTO DI MATEMATICA E INFORMATICA

Corso di Laurea In Informatica
Esame di Programmazione ad Oggetti
16/09/2019
Tempo a Disposizione: 60 min

Cognome: _____ Nome: _____

Matricola: _____ Esercizi lab (quali e non quanti): _____ Numero di Crediti: _____

```
1: class  Franchising : protected list<Sede*> {
2:   public:
3:     Franchising() {};
5:     Franchising(const Franchising& i);
6:     Franchising(const string& n): nome(n){};
6:     void operator=(const Franchising& i);
7:     void aggiungiSede(Tipologia t, const string& nomeSede, const vector<Persona*>& dipendenti);
8:     void rimuoviSede(const string& nomeSede);
9:     void setNome(const string& nome);
10     void trasferisciDipendente(Persona* p, const string& sedeA, const string& sedeB);
11:    ~Franchising();
12:   private:
13:    string nome;
14: };
```

//Implementare quanto riportato in seguito (max 20 punti).
//Una Sede ha un indirizzo e un elenco di dipendenti. Ogni sede ha associata una tipologia (ad esempio: "Principale", "Secondaria", "Magazzino", etc.). Fornire l'intestazione e l'implementazione di almeno tutte le funzioni utilizzate nello svolgimento della classe Sede e della classe SedePrincipale rappresentante la tipologia "Principale".

```
enum Tipologia {Principale=0, Secondaria, Magazzino, Altro};
```


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

Corso di Laurea In Informatica
Esame di Programmazione ad Oggetti
16/09/2019
Tempo a Disposizione: 60 min

```
//Indicare eventuali istruzioni errate e istruzioni mancanti. NB: i metodi utilizzati alla linee 7 e 8 devono essere opportunamente riportati nelle classi Sede e/o SedePrincipale.
int main() {
  1:  list<Franchising*>* f1 = new list<Franchising*>();
  2:  Franchising** f2 = new Franchising*[10];
  3:  f2[0] = new Franchising("McFastFood");
  4:  Franchising* f3 = new Franchising(*f2[0]);
  5:  f1->push_back(f3);
  6:  Franchising* f4 = *(f1->end());
  7:  Sede* s = new SedePrincipale("Roma-ViaRossi-21");
  8:  s->aggiungiDipendente("CiccioPasticcio");
  9:  delete f1->front();
  10: delete f1;
  11: for(unsigned i=0;i<10;++i) { delete f2[i]; };
  12: delete [] f2;
  13: delete f3;
return 0; }
```

Istruzioni Mancanti: _____

Linee Errate: _____

Motivazione: _____

_____

```
void Franchising::aggiungiSede(Tipologia t, const string& nomeSede, const list<Persona*>& dipendenti){/* Se non esiste già una sede con lo stesso nome, aggiunge una nuova sede sulla base dei parametri ricevuti. L'inserimento deve essere tale da rispettare il seguente ordine: prima sono inserite le sedi principali, poi le secondarie, poi i magazzini ed infine le altre sedi. NB: non sono accettate soluzioni che si basano su metodi per l' ordinamento presenti nella libreria STL. */









}
void rimuoviSede(const string& nomeSede){ /* Rimuove, se esiste, la sede con il nome ricevuto come parametro. */




}
Franchising(const Franchising& i){




}
```

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. 16 Settembre 2019

Cognome: _____ Nome: _____

Matricola: _____

Rispondere alle seguenti domande a risposta multipla (2 punti per risposta e motivazione esatta, -1 punti per risposta sbagliata, 0 per risposta non data o risposta esatta ma priva di motivazione):

1. Quale/Quali tra le seguenti implementazioni è/sono corretta/e?

   a)
   ```
   Franchising::~Franchising() {
       for(auto it=this->begin();it!=this->end();++it)
           delete *it;
   }
   ```

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

   c)
   ```
   Franchising::~Franchising() {
       delete [] *this;
   }
   ```

   d) Nessuna delle precedenti. Fornire implementazione:

   Motivazione:

2. Quale/Quali tra le seguenti implementazioni è/sono corretta/e?

   a)
   ```
   void Franchising::trasferisciDipendente(Persona* p, const string& sedeA, const string& sedeB){
       for(auto it=begin();it!=end();it++){
           if(it->getNome()==sedeA && it->haDipendente(p)){
               it->rimuoviDipendente();
           }
           if(it->getNome()==sedeB)
               it->aggiungiDipendente(p);
       }
   }
   ```

   b)
   ```
   void Franchising::trasferisciDipendente(Persona* p, const string& sedeA, const string& sedeB){
       for(auto it=begin();it!=end();it++)
           if((*it)->getNome()==sedeA && (*it)->haDipendente(p))
               (*it)->rimuoviDipendente();
       for(auto it=begin();it!=end();it++)
           if((*it)->getNome()==sedeB)
               (*it)->aggiungiDipendente(p);
   }
   ```

   c)
   ```
   void Franchising::trasferisciDipendente(Persona* p, const string& sedeA, const string& sedeB){
       Persona* p2=new Persona(*p);
       for(auto it=begin();it!=end();it++)
           if((*it)->getNome()==sedeA && (*it)->haDipendente(p)){
               (*it)->rimuoviDipendente();
               delete p;
       }
       for(auto it=begin();it!=end();it++)
           if((*it)->getNome()==sedeB)
               (*it)->aggiungiDipendente(p2);
   }
   ```

d)
```
void Franchising::trasferisciDipendente(Persona* p, const string& sedeA, const string& sedeB){
    for(auto sede:*this)
        if(sede->getNome()==sedeA && sede->haDipendente(p))
            sede->rimuoviDipendente();
    for(auto sede:*this)
        if(sede->getNome()==sedeB)
            sede->aggiungiDipendente(p);
    }
```

Motivazione:

3. Quale/Quali tra le seguenti istruzioni è/sono corretta/e supponendo il metodo `void operator=(const Franchising& i);` essere correttamente implementato.

   a) `Franchising f1,f2; cerr<<(f1=f2);`

   b) `Franchising *f=new Franchising; f=f; delete f;`

   c) `Franchising f1,*f2; f2=&f1;`

   d) Nessuna delle precedenti

   Motivazione:

4. Sia `class Franchising : protected FranchisingDiretto` la definizione di una classe FranchisingDiretto. Quale/i delle seguenti istruzioni è/sono corretta/e in un main?

   a)
   ```
   Franchising* f = new FranchisingDiretto;
   f.pop_back();
   ```

   b)
   ```
   FranchisingDiretto fd;
   Franchising* f = &fd;
   ```

   c)
   ```
   Franchising f;
   FranchisingDiretto* fd = &f;
   ```

   d)
   ```
   Franchising f1, *f2;
   f2->operator=(f1);
   ```

   Motivazione: