UNIVERSITÀ DELLA CALABRIA
DIPARTIMENTO DI MATEMATICA E INFORMATICA

Corso di Laurea In Informatica
Esame di Programmazione ad Oggetti
05/02/2018
Tempo a Disposizione: 60 min

Cognome: _____ Nome: _____

Matricola: _____ Esercizi lab: _____

```
 1: class GestoreRifiuti : protected vector<Rifiuto*>{
 2: public:
 3:    GestoreRifiuti(unsigned c): carico(c), macchine(new MacchinaSmaltimento*[100]),
 4:       numMacchine(0) {}
 5:    GestoreRifiuti(GestoreRifiuti& g);
 6:    GestoreRifiuti& operator=(const GestoreRifiuti& g);
 7:    bool operator==(const GestoreRifiuti& g);
 8:    ~GestoreRifiuti();
 9:   void aggiungiRifiuto(Rifiuto* r);
 10:   bool processaRifiuto();
 11:   void aggiungiMacchinaSmaltimento(MacchinaSmaltimento* m);
 12:   list<MacchinaSmaltimento*> rimuoviMacchineDaRiparare();
 13: private:
 14:   MacchinaSmaltimento** macchine;
 15:   unsigned numMacchine;
 16:   unsigned carico;
};
//Completare opportunamente il main (max 2 punti).
int main() {
    //Un gestore di rifiuti ha un carico massimo (in Kg)
    //di rifiuti che puo' trattare.
    GestoreRifiuti* g1 = new GestoreRifiuti(5000);
    GestoreRifiuti* g2 = new GestoreRifiuti(1000);
    //Una macchina per lo smaltimento puo' essere da riparare o meno.
    MacchinaOrganico m(false);
    MacchinaSmaltimento* pm=&m;
    g1->aggiungiMacchinaSmaltimento(pm);
    //Un rifiuto ha un peso (in Kg).
    Rifiuto* r1 = new RifiutoOrganico(4);
    Rifiuto* r2 = new RifiutoVetro(3);
    g1->aggiungiRifiuto(r1);
    g1->aggiungiRifiuto(r2);
    g2->aggiungiRifiuto(r1);



    return 0;
}
// Implementare i seguenti metodi (max 12 punti).
// Rimuove un rifiuto dalla coda rifiuti e lo processa utilizzando una macchina
appropriata alla tipologia di rifiuto, invocando opportunamente un metodo nella classe
MacchinaSmaltimento se presente tra le macchine disponibili restituendo true.
Altrimenti false.
bool GestoreRifiuti::processaRifiuto(){









}
```

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

```
// Restituisce l'elenco delle macchine per lo smaltimento da riparare e le rimuove tra
le macchine per lo smaltimento disponibili.
list<MacchinaSmaltimento*> GestoreRifiuti::rimuoviMacchineDaRiparare()
{









}
// Una rifiuto può essere aggiunto solo se il suo perso sommato al peso dei rifiuti
precedentemente aggiunti non sfora il carico massimo consentito. Inoltre, i rifiuti
organici hanno maggiore priorità e devono essere inseriti per primi, mentre le altre
tipologie vengono inserite in coda.
void GestoreRifiuti::aggiungiRifiuto(Rifiuto* r)
{









}
// Completare la definizione della seguenti classi (max 4 punti).
class Rifiuto {
protected:
   double peso;









};
class MacchinaSmaltimento {
protected:
   bool daRiparare;









};
```

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. 05 Febbraio 2018

Cognome: _____ Nome: _____

Matricola: _____ Esercizi lab: _____

Rispondere alle seguenti domande a risposta multipla (3 punti per risposta 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)
   ```
   GestoreRifiuti::~GestoreRifiuti() {
       delete [] *this;
       delete [] macchine;
   }
   ```

   b)
   ```
   GestoreRifiuti::~GestoreRifiuti() {
       for(vector<Rifiuto*>:: iterator it=begin(); it!=end();it++)
           delete *it;
       for(unsigned i=0;i<numMacchine;++i)
           delete i;
   }
   ```

   c)
   ```
   GestoreRifiuti::~GestoreRifiuti() {}
   ```

   d) Nessuna delle precedenti. L'implementazione corretta è:

   Motivazione:

2. Quale tra le seguenti implementazioni è corretta?

   a)
   ```
   bool GestoreRifiuti::operator==(const GestoreRifiuti& g){
       if(size()!=g.size()) return false;
       return *this==g;
   }
   ```

   b)
   ```
   bool GestoreRifiuti::operator==(const GestoreRifiuti& g){
       if(size()!=g.size()) return false;
       for(unsigned i=0;i<size();i++)
           if(at(i)!=g[i])
               return false;
       for(unsigned i=0;i<numMacchine;++i)
           if(macchine.at(i)!=g.macchine.at(i))
               return false;
       return true;
   }
   ```

c)
```
bool GestoreRifiuti::operator==(const GestoreRifiuti& g){
    for(unsigned i=0;i<size();i++)
        if(at(i)!=g.at(i))
            return false;
    for(unsigned j=0;j<numMacchine;++j)
        if(macchine[j]!=g.macchine[j])
            return false;
    return true;
}
```

d) Nessuna delle precedenti. L'implementazione corretta è:

Motivazione:

3. Sia **class RifiutoOrganico: protected Rifiuto** la definizione di una classe RifiutoOrganico. Quale tra le seguenti istruzioni **è consentita** nel costruttore di default di RifiutoOrganico?

   a) `cout<<peso<<endl;`

   b) `peso = 10;`

   c) `Sono entrambe consentite`

   d) `Sono entrambe sbagliate`

   Motivazione:

4. Sia **class MacchinaSmaltimentoOrganico: public MacchinaSmaltimento** la definizione di una classe MacchinaSmaltimentoOrganico. Quale tra le seguenti istruzioni è consentita nel main:

   a)
   ```
   MacchinaSmaltimentoOrganico* m;
   m->daRiparare = false;
   ```

   b)
   ```
   MacchinaSmaltimentoOrganico** m=0;
   cout << m->daRiparare <<endl;
   ```

   c) `Sono entrambe consentite`

   d) `Sono entrambe sbagliate`

   Motivazione: