ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ
Одсек за РТИ и Одсек за СИ Београд 15.01.2017.

Испит из Објектно оријентисаног програмирања I

1) *(30 поена)* Одговорити концизно (по једна или две реченице) и прецизно на следећа питања:

а) Да ли тип параметра конструктора класе `X` може бити: (1) `X`, (2)`X*`, (3) `X&`, (4) `X&&`?

б) Како се (већ у време превођења) може спречити стварање објеката неке класе, осим кроз методе изведених класа?

в) Ако постоји општи и неколико специјализованих шаблона класе, којим редоследом се покушава упаривање при генерисању класе из шаблона?

2) *(укупно 70 поена)* Написати на језику *C++* следећи систем класа. Класе опремити оним конструкторима, деструктором и операторима доделе који су потребни за безбедно и ефикасно коришћење класа. Грешке пријављивати изузецима типа једноставних класа које су опремљене писањем текста поруке. За генеричке збирке није дозвољено коришћење класа из стандардне библиотеке шаблона (*STL*).

- *(10 поена)* ***Састојак*** садржи задато име и цену по килограму. Може да се дохвати врста састојка и да се израчуна цена за задату количину састојка у грамима. Уписује се у излазни ток (`it<<s`) у облику *име* **–** *цена***`/kg`**. Врсте ***сланог***, ***слатког*** и ***неутралног*** састојка су **`SLAN`**, **`SLAD`** и **`NEUT`**, респективно.
- *(10 поена)* ***Листа*** садржи произвољан број података неког типа, при чему сваки елемент листе садржи и једну целобројну вредност. Може да се дода један елемент у листу, као и да се дохвати број елемената листе. Може да се постави на први елемент листе, да се прелази на следећи елемент у односу на текући, да се испита да ли постоји текући елемент и да се посебно дохватају податак и целобројна вредност у текућем елементу, са могућношћу промене њихових вредности. Грешка је ако не постоји текући елемент у моменту покушаја дохватања података елемента. Листа не може да се копира ни на који начин, ни иницијализацијом, ни доделом.
- *(15 поена)* ***Јело*** има име и садржи листу састојака (целобројна вредност представља количину састојка за припрему јела изражену у грамима). Цена јела је одређена збиром цене састојака, уз евентуално процентуално повећање или смањење основне цене, које зависи од сата формирања цене јела. Може да се постави податак о процентуалној промени цене јела и податак о сату формирања цене. Може да се дода састојак одређене количине у јело, при чему се пре додавања проверава да ли је састојак адекватан (грешка је ако није), да се дохвати једнословна врста јела и да се израчуна његова цена. Уписује се у излазни ток (`it<<j`) тако што се у првом реду упише *име* **:** *цена*, а у наредним редовима се уписује један састојак по реду у облику *састојак* **:** *количина*. При уништавању јела, уништавају се и сви садржани састојци.
- *(15 поена)* ***Предјело*** и ***главно јело*** су јела која могу да садрже само слане и неутралне састојке и њихове врсте су **`P`** и **`G`**, респективно. Предјело се налази на акцији у периоду 9-12 часова (када се цена умањује за одређени проценат). Цена главног јела се увећава за одређени проценат у периоду 20-23 часа. ***Дезерт*** је јело које може да садржи само слатке и неутралне састојке и његова врста је **`D`**.
- ***Јеловник*** је листа јела, при чему целобројна вредност представља број расположивих порција одређеног јела. При уништавању јеловника, не уништавају се и садржана јела.
- *(15 поена)* ***Генератор*** има задати јеловник и може да се захтева генерисање дневног менија у задатом сату, када се из јеловника на случајни начин бира по једно предјело, главно јело и дезерт. Може да се дохвати цена дневног менија. Уписује се у излазни ток (`it<<g`) тако што се у првом реду испише цена, а затим изабрана јела, свако јело у новом реду.

(5 поена) Написати на језику *C++* **програм** који направи неколико јела, постави им одређени проценат промене цене, дода им састојке, а затим направљена јела дода у јеловник. Потом направи генератор који генерише дневни мени за одређени сат и испише га на стандардном излазу. Користити фиксне параметре – није потребно ништа учитати с главног улаза.

НАПОМЕНЕ: **а)** Испит траје **180** минута.

б) Рад се предаје искључиво у вежбанци за испите (-5 поена за неадекватну вежбанку). Није дозвољено имати поред себе друге листове папира, нити уз себе имати мобилни телефон, без обзира да ли је укључен или искључен.

в) Водити рачуна о уредности. Нечитки делови текста ће бити третирани као непостојећи. Решења задатака навести по горњем редоследу (-1 поен за лош редослед). Препоручује се рад обичном графитном оловком.

г) Решење задатка не треба раздвајати у датотеке. Довољно је за сваку класу навести дефиницију класе и одмах иза ње евентуалне дефиниције метода које нису дефинисане у самој класи.

д) Резултати колоквијума биће објављени на *Web*-у на адреси:
`http://rti.etf.bg.ac.rs/rti/ir2oo1/index.html/`

```
//Sastojak.cpp
#include <string>
#include <iostream>
using namespace std;
class Sastojak {
public:
  enum Vrsta { SLAN, SLAD, NEUT };
  Sastojak(string i, double c)
                      : ime(i), jedCena(c) {}
  virtual Vrsta vrsta() const = 0;
  virtual ~Sastojak() {}
  double cena (double kol) const {
    return jedCena*kol / 1000;
  }
  friend ostream& operator<< (ostream& it,
                          const Sastojak& s) {
    return it << s.ime << " - " <<
                           s.jedCena << "/kg";
}
private: string ime; double jedCena;
};
```

```
//Slan.cpp, Sladak.cpp, Neutralan.cpp
class Slan : public Sastojak {
public:
  Slan(string i,double c) : Sastojak(i, c) {}
  Vrsta vrsta() const override {return SLAN;}
};
class Sladak : public Sastojak {
public:
  Sladak(string i,double c) : Sastojak(i,c)
{}
  Vrsta vrsta() const override {return SLAD;}
};
class Neutralan : public Sastojak {
public:
   Neutralan(string i, double c) :
                            Sastojak(i, c) {}
   Vrsta vrsta() const override{return NEUT;}
};
```

```
//GNemaTek.cpp, GVrsta.cpp
class GNemaTek {
  friend ostream& operator<<(ostream& it,
                          const GNemaTek&) {
    return it << "Nema tekuceg elementa!";
  }
};
class GVrsta {
  friend ostream& operator<<(ostream& it,
                            const GVrsta&) {
    return it << "Losa vrsta sastojka!";
  }
};
```

```
//Lista.cpp
template <typename T>
class Lista {
  struct Elem {
    T pod; int vr; Elem* sled;
    Elem(const T& p, int v, Elem* s=nullptr)
                  : pod(p), vr(v), sled(s) {}
  };
  Elem *prvi, *posl; int br;
  mutable Elem* tek; void brisi();
public:
  Lista() { prvi=posl=tek=nullptr; br = 0; }
  Lista(const Lista &lst) = delete;
  Lista& operator=(const Lista &lst)=delete;
  ~Lista() { brisi(); }
  Lista& dodaj(const T& t, int  v) {
    posl = (!prvi ? prvi : posl->sled) =
                                new Elem(t, v);
    br++; return *this;
  }
  int broj() const { return br; }
  void naPrvi() const { tek = prvi; }
  void naSled() const {
    if (tek) tek = tek->sled;
  }
  bool imaTek() const {return tek!=nullptr;}
  T& dohvPod() const {
    if (!tek) throw GNemaTek();
    return tek->pod;
  }
  int& dohvVr() const {
    if (!tek) throw GNemaTek();
    return tek->vr;
  }
};
template <typename T>
void Lista<T>::brisi() {
  while (prvi) {
    Elem* stari = prvi;
    prvi = prvi->sled; delete stari;
  }
  posl = tek = nullptr;
}
```

```
//Jelo.cpp
class Jelo {
  string ime;
  Lista<Sastojak*> sastojci; void brisi();
protected: double proc; int h;
public:
  Jelo(string i) : ime(i), proc(0), h(0) {}
  void postavi(double p) { proc = p; }
  void postaviH(int hh) { h = hh; }
  void dodaj(Sastojak* s, int kol) {
    if (adekvatan(s)) sastojci.dodaj(s, kol);
    else throw GVrsta();
  }
  virtual bool adekvatan(Sastojak* s)
                                    const = 0;
  virtual char vrsta() const = 0;
  virtual ~Jelo() { brisi(); }
  virtual double cena() const;
  friend ostream& operator<< (ostream& it,
                              const Jelo& j);
};
void Jelo::brisi() {
  for (sastojci.naPrvi(); sastojci.imaTek();
                            sastojci.naSled())
    delete sastojci.dohvPod();
}
double Jelo::cena() const {
    double c = 0;
    for (sastojci.naPrvi();sastojci.imaTek();
                           sastojci.naSled())
      c += sastojci.dohvPod()->cena
                        (sastojci.dohvVr());
    return c;
}
ostream& operator<<(ostream& it,
                                    const Jelo& j) {
  it << j.ime << " : " << j.cena() << endl;
  for (j.sastojci.naPrvi();
  j.sastojci.imaTek(); j.sastojci.naSled()) {
    it << *j.sastojci.dohvPod() << " : " <<
                  j.sastojci.dohvVr() << endl;
  }
  return it;
}
```

```
//Predjelo.cpp, GlavnoJelo.cpp, Dezert.cpp
class Predjelo : public Jelo {
public:
  Predjelo(string i) : Jelo(i) {}
  char vrsta() const override { return 'P'; }
  bool adekvatan(Sastojak* s) const {
    if (s->vrsta() == Sastojak::SLAD)
      return false;
    else return true;
  }
  double cena() const override {
    if (h >= 9 && h <= 12)
      return Jelo::cena()*(100 - proc) / 100;
    else return Jelo::cena();
  }
};
class GlavnoJelo : public Jelo {
public:
  GlavnoJelo(string i) : Jelo(i) {}
  char vrsta() const override { return 'G'; }
  bool adekvatan(Sastojak* s) const {
    if (s->vrsta() == Sastojak::SLAD)
      return false;
    else return true;
  }
  double cena() const override {
    if (h >= 20 && h <= 23)
      return Jelo::cena()*(100 + proc) / 100;
    else return Jelo::cena();
  }
};
class Dezert : public Jelo {
public:
  Dezert(string i) : Jelo(i) {}
  char vrsta() const override { return 'D'; }
  bool adekvatan(Sastojak* s) const {
    if (s->vrsta() == Sastojak::SLAN)
      return false;
    else return true;
  }
};
```

```
//Jelovnik.cpp
class Jelovnik : public Lista <Jelo*> {
public:
  Jelovnik() : Lista<Jelo*>() {}
};
```

```
//Generator.cpp
class Generator {
  Jelovnik* j; double cena; Jelo* jela[3];
public:
  Generator(Jelovnik* jj) : j(jj) {}
  void napravi(int h);
  friend ostream& operator<< (ostream& it,
                          const Generator& g) {
  it << g.cena << endl;
  return it << *g.jela[0] << endl
                          << *g.jela[1] << endl
                          << *g.jela[2] << endl;
  }
};
void Generator::napravi(int h) {
  cena = 0; char vr = 'P';
  for (int i = 0; i < 3; i++) {
    while (1) {
      int broj = rand() / (double)
                  (RAND_MAX + 1.) * j->broj();
      for (j->naPrvi(); --broj >= 0 &&
                    j->imaTek(); j->naSled());
      Jelo* jelo = j->dohvPod();
      int& d = j->dohvVr();
      if (jelo->vrsta() == vr && d > 0) {
        d--; jela[i] = jelo;
        jelo->postaviH(h);
        cena += jelo->cena();
        if (vr == 'P') vr = 'G';
        else vr = 'D';
        break;
      }
    }
  }
}
```

```
//main.cpp
int main() {
  Jelo* j1 = new Predjelo("Sendvic"),
      * j2 = new GlavnoJelo("Piletina sa
                                    sirom"),
      * j3 = new Dezert("Tufahije");
  j1->postavi(10); j2->postavi(20);
  try {
    j1->dodaj(new Neutralan("Hleb",50),200);
    j1->dodaj(new Slan("Sunka", 500), 150);
    j2->dodaj(new Slan("Belo meso",800),300);
    j2->dodaj(new Neutralan("Zejtin", 200),
                                        100);
    j2->dodaj(new Slan("Sir", 400), 100);
    j3->dodaj(new Sladak("Jabuke",300), 200);
    j3->dodaj(new Sladak("Secer", 150), 100);
    j3->dodaj(new Neutralan("Orasi", 1500),
                                        100);
    Jelovnik j; j.dodaj(j1, 5).
                  dodaj(j2, 3).dodaj(j3, 10);
    Generator g(&j); g.napravi(21);
    cout << g << endl;
  }
  catch (GNemaTek g) { cout << g << endl; }
  catch (GVrsta g) { cout << g << endl; }
  catch (...) { cout <<
                    "Doslo je do greske!"; }
  delete j1; delete j2; delete j3; return 0;
}
```

```
// Primer izvrsavanja
670
Sendvic : 85
Hleb - 50/kg : 200
Sunka - 500/kg : 150

Piletina sa sirom : 360
Belo meso - 800/kg : 300
Zejtin - 200/kg : 100
Sir - 400/kg : 100

Tufahije : 225
Jabuke - 300/kg : 200
Secer - 150/kg : 100
Orasi - 1500/kg : 100
```