ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ
Одсек за РТИ и Одсек за СИ Београд 22.01.2016.

Испит из Објектно оријентисаног програмирања I

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

а) За сваки од следећих оператора навести да ли може да се преклопи као глобална пријатељска функција неке класе: `<<` `()` `[]` `==` `=`.

б) Шта исписује следећи програм:

```
#include <iostream>
using namespace std;
class E {static int s; int b=s++; public:
    E(){cout<<"E"<<b<<" ";} ~E(){cout<<"~E"<<b<<" ";} };
int E::s=0;
class O { E e[2]; public: O(){cout<<"O ";} E e1; ~O(){cout<<"~O ";} E e2; };
class I: public O { E e; public: I(){cout<<"I ";} ~I(){cout<<"~I ";} };
int main() {I i;}
```

в) Колико (формалних) параметара може да има специјализовани шаблон (1) класе (2) функције? Да ли специјализована шаблонска класа може да има једнак број шаблонских параметара и аргумената?

2) *(укупно 70 поена)* Реализовати на језику *C++* следећи систем класа. Класе опремити оним конструкторима, деструктором и операторима за доделу вредности, 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*).

- *(15 поена)* ***Боја*** садржи целобројне интензитете црвене, зелене и плаве компоненте, у опсегу од 0 до 255 (грешка је ако вредност компоненте није у том опсегу). Може да се утврди једнакост (==) две боје и да се дохвате све три компоненте боје.
- ***Елемент*** слике се задаје помоћу боје (подразумевано бела боја – све компоненте имају максималну вредност). Може да се дохвати и промени боја елемента слике, као и да се исти упише у излазни ток (<<) у облику **(***црвена***,** *зелена***,** *плава***)**.
- *(15 поена)* ***Слика*** правоугаоног облика садржи матрицу задатог броја врста и колона (подразумевано 3×4) елемената слике задате почетне боје (подразумевано беле). Могу да се дохвате димензије слике, да се приступи елементу са задатим индексима (`sli(i,j)`; грешка је ако је индекс изван опсега), да се промени боја елементу задате врсте и колоне и да се слика упише у излазни ток (<<), свака врста у засебном реду. На ***црно-белој*** слици почетна боја мора бити црна или бела и боја елемената се може променити само у црну или у белу боју. На слици ***у нијанси сиве*** почетна боја мора бити у нијанси сиве и боја елемената се може променити само у боју чије све три компоненте имају исте вредности.
- *(15 поена)* ***Листа*** садржи произвољан број података неког типа. Ствара се празна, након чега се подаци додају појединачно. Не може да се копира ни на који начин. При уписивању у излазни ток (<<) пишу се садржани подаци, сваки у засебном реду. ***Галерија*** је листа слика.
- *(15 поена)* ***Апстрактни филтер*** може да врши обраду задатог елемента слике. Такође, може да се примени над задатом галеријом, када врши обраду сваког елемента сваке слике из галерије. Обрада ***инвертујућег филтра*** састоји се у мењању све три компоненте боје елемента слике, по формули: `novaVrednost=255-staraVrednost`.

(10 поена) Написати на језику *C++* **програм** који створи неколико разнородних слика, промени им боју на појединим елементима, затим створи галерију, у њу дода створене слике и испише је на главном излазу. Потом над свим сликама у галерији примени инвертујући филтер и поново испише галерију. 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`

```
#include <iostream>
using namespace std;
class GBoja {
  friend ostream& operator<<(ostream&
                    it, const GBoja& g) {
    return it << "Neispravna boja!";
  } };
class Boja {
public:
  Boja(int cc=255,int zz=255,int pp=255)
  {
    if (cc<MIN || cc>MAX || zz<MIN
        || zz>MAX || pp<MIN || pp>MAX)
      throw GBoja();
    else {
      c = cc; z = zz; p = pp;
    } }
  int cc() const { return c; }
  int zz() const { return z; }
  int pp() const { return p; }
  friend bool operator==(const Boja& b1,
                         const Boja& b2) {
    return (b1.c == b2.c) && (b1.z ==
                b2.z) && (b1.p == b2.p);
  }
  static const int MIN = 0, MAX = 255;
private: int c, z, p;
};
```

```
class Element {
public:
  Element(Boja b=Boja()):boja(b.cc(),
                         b.zz(),b.pp()) {}
  Boja b() const { return boja; }
  void posBoja(const Boja& b){ boja=b; }
  friend ostream& operator<<(ostream&
                it, const Element& el) {
    return it<<"("<<el.boja.cc()<< ","
 <<el.boja.zz()<<","<<el.boja.pp()<<")";
  }
private: Boja boja;
};
```

```
class GIndeks {
  friend ostream& operator<<(ostream&
                  it, const GIndeks& g) {
    return it << "Indeks van opsega!";
  } };
class Slika {
public:
  Slika(int v=3,int k=4,Boja b=Boja());
  Slika(const Slika& s) { kopiraj(s); }
  Slika(Slika&& s) { premesti(s); }
  ~Slika() { brisi(); }
  Slika& operator=(const Slika& s) {
    if (this!=&s) { brisi();kopiraj(s);}
    return *this; }
  Slika& operator=(Slika&& s) {
    if (this!=&s) {brisi();premesti(s);}
    return *this;
  }
  int v() const { return vrs; }
  int k() const { return kol; }
  Element& operator()(int v, int k) {
    if (v<0||v>=vrs||k<0||k>=kol)
                          throw GIndeks();
    return matrica[v][k];
  }
  const Element& operator()(int v,
                            int k) const {
    return const_cast<Slika&>(*this)
                                    (v, k);
  }
  virtual Slika& posBoja(const Boja& b,
                            int v, int k) {
    (*this)(v,k).posBoja(b);
    return *this;
  }
  friend ostream& operator<<(ostream&
                    it, const Slika& s);
private:
  Boja boja; Element** matrica;
  int vrs, kol;
  void kopiraj(const Slika& s);
  void premesti(Slika& s); void brisi();
};
Slika::Slika(int v,int k,Boja b):vrs(v),
                            kol(k), boja(b) {
  matrica = new Element*[vrs];
  for (int i = 0; i < vrs; i++) {
    matrica[i] = new Element[kol];
    for (int j = 0; j < kol; j++)
      matrica[i][j] = Element(boja);
  }
}
void Slika::premesti(Slika& s) {
  vrs=s.vrs; kol=s.kol; boja=s.boja;
  matrica=s.matrica; s.matrica=nullptr;
}
void Slika::brisi() {
  for (int i = 0; i < vrs; i++)
    delete matrica[i];
  delete []matrica;
}
void Slika::kopiraj(const Slika& s) {
  boja = s.boja;
  matrica = new Element*[vrs = s.vrs];
  for (int i = 0; i < vrs; i++) {
    matrica[i]=new Element[kol=s.kol];
    for (int j = 0; j < kol; j++)
      matrica[i][j]=
                Element(s.matrica[i][j]);
  } }
ostream& operator<<(ostream& it,
                          const Slika& s) {
  for (int i = 0; i < s.vrs; i++) {
    for (int j = 0; j < s.kol; j++)
      it << s.matrica[i][j] << " ";
    it << endl;
  }
  return it;
}
const Boja bela = Boja(bela.MAX,
                    bela.MAX, bela.MAX);
const Boja crna = Boja(crna.MIN,
                    crna.MIN, crna.MIN);
class CrnoBela : public Slika {
public:
  CrnoBela(int v=3,int k=4,
           Boja b = Boja()):Slika(v,k,b){
    if (!(b == bela || b == crna))
      throw GBoja();
  }
  CrnoBela& posBoja(const Boja& b,
                int v, int k) override {
    if ((b == bela) || (b == crna))
      Slika:: posBoja(b, v, k);
    return *this;
   } };
class Siva : public Slika {
public:
  Siva(int v = 3, int k = 4,
           Boja b = Boja()):Slika(v,k,b){
    if (!(b.cc() == b.zz() &&
                          b.zz()==b.pp()))
      throw GBoja();
  }
  Siva& posBoja(const Boja& b, int v,
                        int k) override {
    if (b.cc() == b.zz() &&
                        b.zz() == b.pp())
      Slika:: posBoja(b, v, k);
    return *this;
  } };
```

```
template <typename T>
class Lista {
  void brisi();
public:
  Lista() { prvi = posl = nullptr; }
  Lista(const Lista &lst) = delete;
  ~Lista() { brisi(); }
  Lista& operator=(const Lista &lst)
                                    = delete;
  Lista& operator+=(const T& t) {
    posl=(!prvi?prvi:posl->sled) =
                                new Elem(t);
    return *this; }
  friend ostream& operator<<(ostream&
                        it, const Lista&z) {
    for(Elem* tek=z.prvi; tek;
                          tek=tek->sled)
      it << tek->pod << endl;
    return it;
  }
protected:
  struct Elem {
    T pod; Elem* sled;
    Elem(const T& p,Elem* s = nullptr) {
      pod = p; sled = s; } };
  Elem *prvi, *posl;
};
template <typename T>
void Lista<T>::brisi() {
  while (prvi) {
    Elem* stari=prvi; prvi=prvi->sled;
    delete stari;
  }
  posl = nullptr;
}
class Galerija : public Lista<Slika> {
   friend class Filter;
public:
   Galerija() : Lista<Slika>() {}
};
```

```
class Filter {
public:
  virtual void obrada(Element& el)
                              const = 0;
  void filter(Galerija& g) const;
};
void Filter::filter(Galerija& g) const {
  Galerija::Elem* tek;
  for(tek=g.prvi; tek; tek=tek->sled)
    for(int i=0; i<tek->pod.v(); i++)
      for (int j=0; j<tek->pod.k(); j++)
        obrada(tek->pod(i, j));
}
class Invertor : public Filter {
public:
  void obrada(Element& el) const
                              override {
    Boja b=Boja(b.MAX-el.b().cc(),
   b.MAX-el.b().zz(),b.MAX-el.b().pp());
    el.posBoja(b);
  } };
```

```
int main() {
  try {
    Slika s1(1, 1); CrnoBela cb(1, 1);
    Siva s(1, 1); Galerija g;
    s1.posBoja(Boja(50, 20, 200), 0, 0);
    cb.posBoja(Boja(0, 0, 0), 0, 0);
    s.posBoja(Boja(100, 100, 100),0,0);
    ((g+=s1)+=cb)+=s;
    cout << g; Invertor i; i.filter(g);
    cout << g;
  }
  catch (GIndeks& g) { cout << g; }
  catch (GBoja& g) { cout << g;  }
  catch (...) {}
  return 0; }
```

```
(50,20,200)
(0,0,0)
(100,100,100)
(205,235,55)
(255,255,255)
(155,155,155)
```