ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ
Одсеци ИР, ОТ, ОС и ОГ Београд 14.01.2018.

Завршни колоквијум из Објектно оријентисаног програмирања I

1) *(укупно 100 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*).

- *(20 поена)* Двострано уланчана ***листа*** садржи произвољан број података неког типа. Може да се дода елемент на почетак или на крај листе. Текућим елементом може да се прогласи први или последњи елемент листе, може да се прелази на следећи или претходни елемент у односу на текући, као и да се испита да ли постоји текући елемент. Додавање елемента не мења текући елемент. Може да се дохвати податак у текућем елементу (са могућношћу промене његове вредности) и да се уклони текући елемент из листе, при чему претходни елемент (ако постоји) постаје текући. Грешка је ако не постоји текући елемент у моменту покушаја дохватања података или уклањања елемента из листе. Листа не може да се копира ни на који начин.
- *(30 поена)* ***Адреса*** *(MAC Address)* садржи низ од шест бајтова чија се вредност задаје при стварању. Подразумевана вредност им је нула. Може да се испита да ли су две адресе једнаке (`a1==a2`). Адреса се уписује у излазни ток (`it<<adresa`) у облику *hex*`:`*hex*`:`*hex*`:`*hex*`:`*hex*`:`*hex* (где *hex* означава вредност бајта у хексадецималном бројном систему).
- ***Оквир податка*** садржи изворишну адресу, одредишну адресу, садржај који може имати произвољан број бајтова и величину садржаја изражену у броју бајтова. Све вредности се задају при стварању и могу да се дохвате. Оквир не може да се копира ни на који начин. Оквир се уписује у излазни ток (`it<<okvir`) у облику **`Frame[src:`** *izv_adr*`,` **`dst:`***odr_adr*`,` **`content:`***sadržaj***`]`**.
- ***Мрежни интерфејс*** садржи адресу, информацију о интерфејсу са друге стране са којим је директно повезан и информацију ком мрежном уређају припада. Припадност мрежном уређају се задаје приликом стварања, док се остала поља накнадно постављају и дохватају. Интерфејс може да прихвати оквир и том приликом га безусловно прослеђује свом мрежном уређају на даљу обраду. Интерфејс може да пошаље оквир ка интерфејсу са друге стране са којим је директно повезан.
- *(40 поена)* ***Мрежни уређај*** садржи задати број мрежних интерфејса. Може да се дохвати слободан тј. неповезан интерфејс. Мрежни уређај врши обраду оквира доспелих преко његових интерфејса. ***Хаб***, ***свич*** и ***хост*** су мрежни уређаји. ***Хаб*** приликом обраде оквира не узима у обзир његову одредишну адресу већ примљени оквир шаље на све своје интерфејсе, осим на интерфејс преко којег је дати оквир примљен. ***Свич*** приликом обраде оквира узима у обзир одредишну адресу оквира. Свич одржава интерну табелу реализовану као двострано уланчана листа, чији елементи садрже показиваче на парове (*адреса*, *интерфејс*), тако што за сваки примљени оквир памти његову изворишну адресу и интерфејс преко којег је примљен. Свич прослеђује оквир на основу садржаја ове табеле и одредишне адресе датог оквира. У случају да за одредишну адресу неког оквира не постоји улаз у табели свича не врши се прослеђивање оквира. ***Хост*** има аутоматски генерисан јединствени идентификатор. Може да пошаље оквир на основу задатих података за формирање оквира. Приликом обраде оквира врши се испис у излазни ток облика **`Host:`***id*`-`*okvir* **`allow`** ако је оквир намењен датом хосту, односно **`Host:`***id*`-`*okvir* **`deny`** ако оквир није намењен датом хосту (пореди се одредишна адреса и адреса интерфејса хоста преко којег је примљен оквир).

(10 поена) Написати на језику *C++* програм који направи два хоста, један хаб и потребан број интерфејса, додели адресе интерфејсима, изврши повезивање одговарајућих интерфејса и пошаље оквир са једног ка другом хосту. Користити фиксне параметре - није потребно ништа учитавати са главног улаза.

НАПОМЕНЕ:

а) Колоквијум траје **150** 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>
#include <string>
using namespace std;
class GNemaTek {
  friend ostream& operator<<(ostream &it,
                         const GNemaTek&){
    return it << "Nema tekuceg elementa!";
  }
};
```

```
template <typename T>
class Lista {
  struct Elem {
    T pod; Elem *pret, *sled;
    Elem(const T &t, Elem *p, Elem *s) :
              pod(t), pret(p), sled(s) {}
  };
  Elem *prvi, *posl; mutable Elem *tek;
  void obrisiListu();
public:
  Lista() { prvi = posl = tek = nullptr;}
  Lista(const Lista &) = delete;
  Lista& operator=(const Lista &) = delete;
  ~Lista() { obrisiListu(); }
  Lista& dodajKraj(const T &t) {
    posl = (!prvi? prvi: posl->sled)
         = new Elem(t, posl, nullptr);
    return *this;
  }
  Lista& dodajPocetak(const T &t) {
    prvi = (!prvi? posl: prvi->pret)
         = new Elem(t, nullptr, prvi);
    return *this;
  }
  void naPrvi() const { tek = prvi; }
  void naPoslednji() const { tek = posl; }
  void naSled() const {
    if (tek) tek = tek->sled;
  }
  void naPret() const {
    if (tek) tek = tek->pret;
  }
  bool imaTek() const {
    return tek != nullptr;
  }
  T& dohvTek() const {
    if (!tek)
      throw GNemaTek();
    return tek->pod;
  }
  Lista& obrisiTek();
};
template <typename T>
void Lista<T>::obrisiListu() {
  while (prvi) {
    Elem *stari = prvi;
    prvi = prvi->sled;
    delete stari;
  }
  posl = tek = nullptr;
}
template <typename T>
Lista<T>& Lista<T>::obrisiTek() {
  if (!tek)
    throw GNemaTek();
  if (tek->sled)
    tek->sled->pret = tek->pret;
  if (tek->pret)
    tek->pret->sled = tek->sled;
  if (tek == posl) posl = posl->pret;
  if (tek == prvi) prvi = prvi->sled;
  Elem *stari = tek; tek = tek->pret;
  delete stari; return *this;
}
```

```
class Adresa {
  unsigned char adr[6] = {0,0,0,0,0,0};
public:
  Adresa() = default;
  Adresa(const unsigned char *nizAdr) {
    for (int i = 0; i < 6; i++)
      adr[i] = nizAdr[i];
  }
  friend bool operator==(
   const Adresa &a1, const Adresa &a2) {
    for (int i = 0; i < 6; i++)
      if (a1.adr[i] != a2.adr[i])
        return false;
    return true;
  }
  friend ostream& operator<<(ostream &it,
                      const Adresa &mac) {
    for (int i = 0; i < 5; i++)
      it << hex << (int) mac.adr[i] << ":";
    return it << hex << (int) mac.adr[5];
  }
};
```

```
class Okvir {
  Adresa izvMAC, odrMAC;
  unsigned char *sadrzaj; int vel;
public:
  Okvir(const Adresa &izvMAC,
        const Adresa &odrMAC,
        unsigned char *s, int v) :
          izvMAC(izvMAC), odrMAC(odrMAC),
          sadrzaj(s), vel(v) {}
  Okvir(const Okvir&) = delete;
  Okvir& operator=(const Okvir&) = delete;
  Adresa dohvIzvMAC() const { return izvMAC; }
  Adresa dohvOdrMAC() const { return odrMAC; }
  unsigned char* dohvSadrzaj() const {
    return sadrzaj;
  }
  int dohvVel() const { return vel; }
  friend ostream& operator<<(ostream &it,
                             const Okvir &o) {
    it << "Frame[src:" << o.izvMAC <<
      ", dst:" << o.odrMAC << ", content:";
    for (int i = 0; i < o.vel; i++)
      it << o.sadrzaj[i];
    return it << "] ";
  }
};
```

```
class Interfejs {
protected:
  Adresa macAdr; Interfejs *sused = nullptr;
  MrezniUredjaj *mojUredjaj = nullptr;
public:
  Interfejs(MrezniUredjaj *u) : mojUredjaj(u){}
  Interfejs(const Interfejs &) = delete;
  Interfejs(Interfejs &&) = delete;
  Adresa dohvMAC() const {return macAdr;}
  void postMAC(const Adresa &mac) {
    macAdr = mac;
  }
  Interfejs* dohvSused() const {return sused;}
  void postSused(Interfejs *s) {sused = s;}
  void posalji(const Okvir&) const;
  void prihvati(const Okvir&) const;
};
void Interfejs::posalji(const Okvir &o) const{
  if (sused)
    sused->prihvati(o);
}
void Interfejs::prihvati(const Okvir &o) const{
  mojUredjaj->obradi(o, this);
}
```

```
class MrezniUredjaj {
protected:
  Interfejs **ints; int intsNum;
public:
  MrezniUredjaj(int n) : intsNum(n) {
    ints = new Interfejs* [intsNum];
    for (int i = 0; i < intsNum; i++)
      ints[i] = new Interfejs(this);
  }
  MrezniUredjaj(const MrezniUredjaj &) = delete;
  MrezniUredjaj(MrezniUredjaj &&) = delete;
  virtual ~MrezniUredjaj() {
    for (int i = 0; i < intsNum; i++)
      delete ints[i];
    delete[] ints;
  }
  Interfejs* dohvSlobodan() const {
    for (int i = 0; i < intsNum; i++)
      if (ints[i]->dohvSused() == nullptr)
        return ints[i];
    return nullptr;
  }
  virtual void obradi(const Okvir&,
     const Interfejs*) = 0;
};
```

```
class Hab : public MrezniUredjaj {
public:
  using MrezniUredjaj::MrezniUredjaj;
  virtual void obradi(const Okvir &okvir,
     const Interfejs *ulazInt) override {
    for (int i = 0; i < intsNum; i++)
      if(ints[i] != ulazInt)
        ints[i]->posalji(okvir);
  }
};
```

```
class Svic : public MrezniUredjaj {
  struct Par {
    const Adresa macAdr;
    const Interfejs *interfejs;
    Par(Adresa mac, const Interfejs *i) :
        macAdr(mac), interfejs(i) {}
  };
  Lista<Par*> tab;
public:
  using MrezniUredjaj::MrezniUredjaj;
  ~Svic() override {
    for(tab.naPrvi();tab.imaTek();tab.naSled()){
      delete tab.dohvTek();
      tab.dohvTek() = nullptr;
    }
  }
  virtual void obradi(const Okvir &o,
            const Interfejs *ulazInt) override {
    bool noviPar = true;
    const Interfejs *tempInt = nullptr;
    for (tab.naPrvi();tab.imaTek();tab.naSled())
      if(tab.dohvTek()->macAdr==o.dohvIzvMAC()) {
        if(tab.dohvTek()->interfejs == ulazInt) {
          noviPar = false; break;
        }
        else
          tab.obrisiTek();
      }
    if (noviPar == true)
      tab.dodajKraj(new
            Par(o.dohvIzvMAC(), ulazInt));
    for (tab.naPrvi();tab.imaTek();tab.naSled())
      if(tab.dohvTek()->macAdr==o.dohvOdrMAC()) {
        tab.dohvTek()->interfejs->posalji(o);
        break;
    }
  }
};
```

```
class Host : public MrezniUredjaj {
  static int tekId; int mojId = ++tekId;
public:
  using MrezniUredjaj::MrezniUredjaj;
  virtual void obradi(const Okvir &o,
          const Interfejs *ulazInt) override {
    cout << "Host: " <<  mojId << " - " << o;
    if (o.dohvOdrMAC() == ulazInt->dohvMAC())
      cout << "ALLOW" << endl;
    else
      cout << "DENY" << endl;
  }
  bool posaljiOkvir(const Okvir &o) {
    for (int i = 0; i < intsNum; i++)
      if (ints[i]->dohvMAC() ==
                    o.dohvIzvMAC()) {
        ints[i]->posalji(o);
        return true;
      }
    return false;
  }
};
int Host::tekId = 0;
```

```
void main() {
  try {
    unsigned char macNizA[6] =
                        {1, 1, 1, 1, 1, 1};
    unsigned char macNizB[6] =
                        {2, 2, 2, 2, 2, 2};
    Adresa macAdrA(macNizA);
    Adresa macAdrB(macNizB);
    Host *a = new Host(1), *b = new Host(1);
    Hab *hab = new Hab(2);
    Interfejs *habInt1 = hab->dohvSlobodan();
    Interfejs *aInt = a->dohvSlobodan();
    aInt->postMAC(macAdrA);
    aInt->postSused(habInt1);
    habInt1->postSused(aInt);
    Interfejs *habInt2 = hab->dohvSlobodan();
    Interfejs *bInt = b->dohvSlobodan();
    bInt->postMAC(macAdrB);
    bInt->postSused(habInt2);
    habInt2->postSused(bInt);
    unsigned char cont[3] = { 'm','s','g' };
    Okvir okvir(macAdrA, macAdrB, cont, 3);
    a->posaljiOkvir(okvir);
    delete a;
    delete b;
    delete hab;
  }
  catch(GNemaTek g) {
    cout << g;
  }
}
```

//primer izvršavanja:

Host: 2 - Frame[src:1:1:1:1:1:1, dst:2:2:2:2:2:2, content:msg] ALLOW