ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ
Одсек за СИ Београд 06.12.2018.

Други колоквијум из
Објектно оријентисаног програмирања I

1) *(укупно 70 поена)* Написати на језику C++ следеће класе (класе опремити оним конструкторима, деструктором и операторима доделе који су потребни за безбедно и ефикасно коришћење класа у општем случају):

- *(15 поена)* Непопуњен ***пар*** има једнозначну, аутоматски генерисану целобројну шифру, текстуални опис, низ од 3 реалне вредности, које представљају квоте за победу домаћина, нерешен исход и победу госта, респективно, као и стварни исход утакмице. Иницијални стварни исход је *NEODREDJEN*. Подаци пара, осим стварног исхода и шифре, се задају при стварању (није потребна провера улазних података). Могуће је дохватити квоту (`par[ishod]`) за одређени могући исход (*JEDAN, IKS, DVA*). Могуће је поставити стварни исход пара (`par(ishod)`) и дохватити стварни исход (`par()`). Пар може да се упише у излазни ток (`it<<par`) у облику *шифра* **`-`** *опис* **`(`***квота_један***`,`** *квота_икс***`,`** *квота_два***`)`**. Уколико стварни исход није *NEODREDJEN*, испред квоте стварног исхода треба уметнути знак **`T`**. Пар не може да се копира ни на који начин.
- *(30 поена)* ***Тикет*** се ствара са задатом реалном уплатом и садржи произвољан број непопуњених парова са придруженим (попуњеним) жељеним исходима. Ствара се празан, након чега парови могу да се додају појединачно, заједно са жељеним исходом. Може да се дохвати вредност уплате на тикету. Може да се израчуна вредност исплате тикета (`*tiket`). Може да се одреди да ли је тикет добитан (`~tiket`). Може да се направи копија тикета. Тикет може да се упише у излазни ток (`it<<tiket`) у облику **`Tiket -`** *статус_добитка* **`:`** **`(`***уплата***`)`** *вредност_исплате*, након чега се у засебним редовима исписују појединачни парови и жељени исходи у формату *пар* **`|`** *жељени_исход*.
- *(10 поена)* ***Обичан*** тикет је добитан уколико за све парове на тикету важи да су жељени исход и стварни исход једнаки. Вредност исплате обичног тикета одређује се као производ квота жељених исхода на свим паровима тикета и реалне уплате на тикету и то само ако је тикет добитан. У супротном, вредност исплате је 0.
- *(15 поена)* ***Кладионица*** се ствара празна, задатог назива и садржи произвољан број тикета. Тикет може да се одигра у кладионици, тако што се његова копија дода кладионици (`kladionica+=tiket`). Може да се одреди тренутна зарада кладионице као разлика суме свих уплата на свим тикетима и суме добитних тикета (`++kladionica`). Кладионица се уписује у излазни ток (`it<<kladionica`) у облику *назив* **`:`** *тренутна_зарада*, након чега се у засебним редовима исписују појединачни тикети. Кладионица не може да се копира ни на који начин.

(0 поена) Приложена је главна функција која ствара једну кладионицу и неколико парова, а затим направи неколико тикета у које дода по неколико парова са жељеним исходима и те тикете одигра у кладионици. Након тога постави коначне исходе свих парова и испише кладионицу на главном излазу. Називе класа и метода прилагодити приложеној главној функцији, тако да се програм може превести и извршити.

НАПОМЕНЕ: **а)** Израда задатка траје **100** минута.
б) Рад се предаје на предвиђеном мрежном диску.
в) На располагању је документација на *Web*-у **на адресама: http://www.cplusplus.com/ и http://en.cppreference.com**
г) Није дозвољено уз себе имати електронске уређаје, без обзира да ли су укључени или искључени.
д) Резултати колоквијума биће објављени на *Web*-у на адреси: `http://rti.etf.rs/rti/ir2oo1/index.html/`

```
#include <iostream>
#include <string>
using namespace std;
class Par {
public:
   enum Ishod {JEDAN=1,IKS=0,DVA=2,NEODREDJEN};
private:
   static int sifraS;
   int sifra = ++sifraS;
   string opis;
   double kvote[3];
   enum Ishod ishod = NEODREDJEN;
public:
   Par(string o, double k[3]) : opis(o) {
      for (int i = 0; i < 3; i++) kvote[i]=k[i];}
   Par(const Par&) = delete;
   Par& operator=(const Par&) = delete;
   double operator[](Ishod ishod) const {
      return ishod!=NEODREDJEN?kvote[ishod]:0; }
   void operator()(Ishod i) { ishod = i; }
   Ishod operator()() const { return ishod;  }
   friend ostream& operator<<(ostream& os,
   const Par &p) {
      return os << p.sifra << " - " << p.opis <<
      " (" << (p.ishod == JEDAN? "T" : "") <<
      p.kvote[JEDAN] << ", " << (p.ishod == IKS?
      "T" : "") << p.kvote[IKS] << ", " <<
      (p.ishod == DVA ? "T" : "") << p.kvote[DVA]
      << ")"; }
};
```

```
class Tiket {
   void brisi();
   void kopiraj(const Tiket &t);
   void premesti(Tiket &t) {
      prvi=t.prvi; posl=t.posl; uplata=t.uplata;
      t.prvi = t.posl = nullptr; }
protected:
   struct Elem {
      const Par &par;
      Par::Ishod ishod;
      Elem *sled = nullptr;
      Elem(const Par &par, Par::Ishod ishod) :
      par(par), ishod(ishod) {}
   };
   Elem *prvi = nullptr, *posl = nullptr;
   double uplata;
public:
   Tiket(double uplata) : uplata(uplata) {}
   Tiket(const Tiket &t) { kopiraj(t);  }
   Tiket(Tiket &&t) { premesti(t); }
   Tiket& operator=(const Tiket &t) {
      if (this != &t) { brisi(); kopiraj(t); } }
   Tiket& operator=(Tiket &&t) {
      if (this != &t) { brisi(); premesti(t); } }
   virtual ~Tiket() { brisi(); }
   Tiket& dodajPar(const Par &par,
                   Par::Ishod ishod) {
      Elem *novi = new Elem(par, ishod);
      posl = (prvi ? posl->sled : prvi) = novi;
      return *this; }
   double dohvUplata() const { return uplata; }
   virtual double operator*() const = 0;
   virtual bool operator~() const = 0;
   virtual Tiket* kopija() const = 0;
   friend ostream& operator<<(ostream &os,
                              const Tiket &t);
};
void Tiket::brisi() {
   while (prvi) { Elem *stari = prvi;
      prvi = prvi->sled; delete stari; }
   posl = nullptr; }
void Tiket::kopiraj(const Tiket &t) {
   for (Elem *tek=t.prvi; tek; tek=tek->sled) {
      Elem *novi=new Elem(tek->par, tek->ishod);
      posl = (prvi ? posl->sled : prvi) = novi;}
   uplata = t.uplata; }
ostream& operator<<(ostream &os,const Tiket &t){
   os << "Tiket - " << ~t << " : (" << t.uplata
                          << ")" << *t << endl;
   for(Tiket::Elem*tek=t.prvi;tek;tek=tek->sled)
   { os<<tek->par<<" | "<<tek->ishod<<endl; }
   return os; }
```

```
class Obican : public Tiket {
public: using Tiket::Tiket;
   double operator*() const override;
   bool operator~() const override;
   Obican* kopija() const override {
      return new Obican(*this); }
};
double Obican::operator*() const {
   if (~*this) { double v = 1;
      for (Elem *tek=prvi; tek; tek=tek->sled)
         v *= tek->par[tek->ishod];
      return v * uplata;
   } else return 0; }
bool Obican::operator~() const {
   for (Elem *tek=prvi; tek; tek=tek->sled)
      if (tek->par() != tek->ishod) return false;
      return true; }
```

```
class Kladionica {
   struct Elem_T {
      Tiket *tiket;
      Elem_T *sled = nullptr;
      Elem_T(const Tiket &t):tiket(t.kopija()){}
      ~Elem_T() { delete tiket; }
   };
   Elem_T *prviT = nullptr, *poslT = nullptr;
   string naziv;
public:
   Kladionica(string naziv) : naziv(naziv) {}
   Kladionica(const Kladionica &) = delete;
   Kladionica& operator=(Kladionica &) = delete;
   Kladionica& operator+=(const Tiket &t) {
      Elem_T *novi = new Elem_T(t);
      poslT=(prviT ? poslT->sled : prviT)=novi;
      return *this; }
   double operator++() const;
   friend ostream& operator<<(ostream &os,
        const Kladionica &k);
   ~Kladionica();
};
double Kladionica::operator++() const {
   double zarada = 0;
   for (Elem_T *tek=prviT; tek; tek=tek->sled)
      zarada += tek->tiket->dohvUplata() -
                           *(*tek->tiket);
   return zarada; }
ostream& operator<<(ostream &os, const
                                 Kladionica &k) {
   os << k.naziv << " : " << ++k << endl;
   for (Kladionica::Elem_T *tek = k.prviT; tek;
                                 tek=tek->sled)
      os << *(tek->tiket) << endl;
   return os; }
Kladionica::~Kladionica(){
   while (prviT) { Elem_T *s=prviT;
      prviT=prviT->sled; delete s;} }
```

```
int Par::sifraS = 0;
int main() {
   Kladionica k("MaleroznaKladionica");
   double kvote3[3] = { 2.55, 3.15, 2.85 };
   Par par3("New - Whu", kvote3);
   double kvote4[3] = { 4.2, 3.5, 1.9 };
   Par par4("Sou - Utd", kvote4);
   Obican t1(500); t1.dodajPar(par3, Par::DVA);
   Obican t2(100);
   t2.dodajPar(par3, Par::JEDAN).dodajPar(par4,
   Par::IKS); k += t1; k += t2;
   par3(Par::JEDAN); par4(Par::IKS); cout << k;
}
```

```
//Primer ispisa
MaleroznaKladionica : -723
Tiket - 0 : (500)0
1 - New - Whu (T3.15, 2.55, 2.85) | 2

Tiket - 1 : (100)1323
1 - New - Whu (T3.15, 2.55, 2.85) | 1
2 - Sou - Utd (3.5, T4.2, 1.9) | 0
```