
Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Objektno orijentisano programiranje
(OF2OO1, OS2OO1, OS3OOP, OE2OOP, OT2OOP)

Nastavnik: doc. dr Dragan Milićev

Ispitni rok: Februar 2007.

Datum: 7.3.2007.

Kandidat: _____

Broj Indeksa: _____ *E-mail:* _____

Ispit traje 3 sata. Dozvoljeno je korišćenje literature.

Pismeni ispit:

Zadatak 1 _____/10

Zadatak 2 _____/10

Zadatak 3 _____/10

Zadatak 4 _____/10

Zadatak 5 _____/20

Domaći zadaci:

Obavezni deo _____/40

Neobavezni deo _____/10

Ukupno na ispitu: _____/60

Ukupno na domaćem: _____/50

Ukupno: _____/110

Ocena: _____ (_____)

Napomena: Ukoliko u zadatku nešto nije dovoljno precizno definisano, student treba da uvede razumnu pretpostavku, da je uokviri (da bi se lakše prepoznala prilikom ocenjivanja) i da nastavi da izgrađuje preostali deo svog odgovora na temeljima uvedene pretpostavke. Ocenjivanje unutar potpitanja je po sistemu "sve ili ništa", odnosno nema parcijalnih poena. Kod pitanja koja imaju ponuđene odgovore treba **samo zaokružiti** jedan odgovor. Na ostala pitanja odgovarati **čitko, kratko i precizno**. Zadaci 1-4 su eliminatorni prema Pravilima predmeta.

1. (10 poena)

Dat je deo koda na jeziku C++. Za svaku liniju označenu brojevima 1-10 navesti da li je ispravna (prevodilac neće prijaviti grešku) ili nije (prevodilac će prijaviti grešku) – u tabelu upisati „Da“ ili „Ne“.

```
class X {
private:
    X (const X&);
};

class Y {
private:
    Y& operator= (const Y&);
};

class Z : public X {};

void main () {
    X x;      // 1
    Y y;      // 2
    Z z;      // 3
    X x1=x;   // 4
    Y y1=y;   // 5
    Z z1=z;   // 6
    x1=x;     // 7
    y1=y;     // 8
    z1=z;     // 9
    x=z;      // 10
};
```

Odgovor:

Linija	Ispravna
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

2. (10 poena)

Dat je sledeći program na jeziku C++:

```
#include <iostream>
using namespace std;

class X {
public:
    X () : myID(id) {}
    X (const X& right) : myID(++id) {}
    friend ostream& operator<< (ostream&, const X&);
private:
    static int id;
    int myID;
};

int X::id = 0;

ostream& operator<< (ostream& os, const X& x) {
    return os<<x.myID;
}

X f (X x1, X& x2) {
    x1 = x2;
    return x2;
}

void main () {
    X x1, x2 = x1;
    X x3 = f(x1,x2);
    cout<<x1<<' ' <<x2<<' ' <<x3;
}
```

Šta ispisuje ovaj program (odgovor ispisati tačno kako i program ispisuje)?

Odgovor:

NAPOMENA: Zbog optimizacija koje primenjuju neki prevodioci, može se desiti da se ispiše 0,1,3, umesto očekivanih 0,1,4. Objašnjenje: ako je rezultat funkcije kompletan privremeni objekat, a ne referenca na neki objekat, i taj rezultat se koristi kao argument konstruktora kopije, neće se pozvati konstruktor kopije, već će kao podatak koji se pozivom konstruktora kopije stvara biti upotrebljen upravo taj privremeni objekat.

3. (10 poena)

Na jeziku C++ realizovati klasu `point` koja apstrahuje tačku u Dekartovom koordinatom sistemu. Tačka se predstavlja koordinatama (x,y) tipa `double`. Ova klasa treba da obezbedi sledeće:

- Inicijalizaciju koordinatama, pri čemu je podrazumevana inicijalizacija na $(0,0)$.
- Operacije poređenja na jednakost (`operator==`) i nejednakost (`operator!=`) dve tačke.
- Operaciju izračunavanja rastojanja (`operator-`) dve tačke.
- Operaciju koja vraća tačku simetričnu datoj tački u odnosu na y -osu (`operator-`).
- Operaciju koja vraća tačku simetričnu datoj tački u odnosu na x -osu (`operator!`).
- Operaciju pomeranja tačke za $(\Delta x, \Delta y)$.

4. (10 poena)

Potrebno je realizovati sledeći sistem klasa i njihovih relacija. *Građanin (Citizen)* može imati jedan ili više svojih *Ličnih Dokumentata (Personal Document)*, koji mogu biti *Pasoš (Passport)* ili *Lična Karta (ID Card)*. Svaki Lični Dokument ima svoj datum izdavanja (*date of issue*) i datum isteka važnosti (*date of expiry*).

- (a) Nacrtati dijagram klasa koji prikazuje UML model opisanog sistema.
- (b) Napisati kompletan C++ kod klase *Građanin (Citizen)* i svih njenih operacija, uključujući operaciju `getValidPassport()` koja vraća Pasoš datog građanina koji je još uvek važeći, ako takvog ima. Pretpostaviti da je na raspolaganju apstraktni tip podataka *Datum (Date)* sa svim potrebnim operacijama hronološkog poređenja.

5. (20 poena) Konstruktivni zadatak

Potrebno je napraviti sledeće proširenje TSS: dodati podršku za ponašanje raskrsnice na kojoj radi saobraćajni policajac. Posao policajca se sastoji u sledećem: sva vozila koja uđu u raskrsnicu treba da odu u jednu od N jednosmernih ulica, a policajac šalje vozilo u onu ulicu u kojoj se u datom trenutku nalazi najmanje vozila.

Pri realizaciji zadatka, pri korišćenju postojećeg programskog koda TSS, jasno naznačiti koje se klase i metode TSS koriste, kao i eventualne izmene nad njima.

- a) (10 poena) Klasa **Raskrsnica** i potrebne izmene u postojećim klasama.
- b) (5 poena) Dodati u postojeće **visitor** klase metode odgovarajuće klasi **Raskrsnica**.
- c) (5 poena) Napisati programski kod koji stvara i ispravno povezuje 1 izvor, 1 raskrsnicu sa 4 ulice i potrebnim brojem ponora, te pokreće simulaciju i ispisuje rezultate simulacije (za raskrsnicu i za ulice obavezno ispisati broj vozila). Na kraju, dealocirati korišćenu dinamičku memoriju.