Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Objektno orijentisano programiranje
(OE2OOP, OE4OOP, OF2OO1, OS2OO1, OS3OOP, OT2OOP, OT3OOP)
Nastavnik: Prof. dr Dragan Milićev
Ispitni rok: Februar 2009.
Datum: 13.02.2009.

Kandidat: _____

Broj Indeksa: _____ *E-mail:*_____

*Ispit ima 2 dela ukupnog trajanja 3 sata. Na prvom delu **nije** dozvoljeno korišćenje literature. Na drugom delu **jeste** dozvoljeno korišćenje literature. Trajanje prvog dela student određuje prema ličnom nahođenju. Student može početi da koristi literaturu kad preda prvi deo.*

UPISATI SVOJE PODATKE I NA PRVOJ STRANI DRUGOG DELA ISPITA!

Pismeni ispit:		***Rad u toku semestra:***	
Zadatak 1	_____/10	*Projekat*	_____/25
Zadatak 2	_____/10	*Domaći zadaci*	_____/25
Zadatak 3	_____/10		
Zadatak 4	_____/10		
Zadatak 5	_____/10		
Ukupno na ispitu:	_____**/50**	***Ukupno u toku semestra:***	_____**/50**

Ukupno: _____/100

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**.

1. (10 poena)

Dati su sledeći delovi koda na jeziku C++:

(a)
```
double sqrt(double);
int a = 0, b = a+2;
double c = 0;
(c + a++) = sqrt(b);
```

(b)
```
class B {};

class D : public B {
public: D (int) {}
};
```

(c)
```
class B {
protected: B(int);
};

class D : public B {
public: D (int) {}
};
```

(d)
```
class X;
X* p = new X;
```

(e)
```
class X {};
X* p = new X;
```

Date su sledeće poruke prevodioca o greškama:
(P1) "Syntax error" ("Greška u sintaksi")
(P2) "Identifier not declared" ("Identifikator nije deklarisan")
(P3) "Class not defined" ("Klasa nije definisana")
(P4) "The operand is not an lvalue" ("Operand nije lvrednost")
(P5) "No appropriate conversion" ("Ne postoji odgovarajuća konverzija")
(P6) "No appropriate constructor" ("Ne postoji odgovarajući konstruktor")
(P7) "The constructor is not accessible" ("Konstruktor nije dostupan")
U sledeću tabelu upisati oznake poruka o greškama prevodioca (P1 do P7) za svaki dati deo koda. Ukoliko je dati deo koda ispravan, upisati oznaku "OK". Napomena: moguće je da neki dati deo koda ima više grešaka; u tom slučaju u tabelu upisati oznake svih poruka koje odgovaraju tim greškama. Napomena: prikazani delovi koda nisu u istoj datoteci.

Deo koda	**Poruke o greškama**
a	
b	
c	
d	
e	

2. (10 poena)

Dat je sledeći deo koda:

```
class Base {
public:
  virtual void f();
  virtual void g();
           void h();
};

class Derived : public Base {
public:
  virtual void f();
           void h();
};
```

Navesti koju metodu poziva svaka od dole označenih linija koda 1-11. Odgovor dati upisom konkretne metode sa punim imenovanjem, npr. `Base::f()` ili `Derived::h()` itd. u dole datu tabelu.

```
void main () {
  Base b; Derived d;
  Base* pb1 = &b;
  Base* pb2 = &d;
  Derived* pd = &d;

  pb1->f();  // 1
  pb2->f();  // 2
  pd->f();   // 3

  pb1->g();  // 4
  pb2->g();  // 5
  pd->g();   // 6

  pb1->h();  // 7
  pb2->h();  // 8
  pd->h();   // 9

  Base& b2 = d;
  b2.f();    // 10
  Base b3 = d;
  b3.f();    // 11
}
```

Linija koda	Metoda
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	

3. (10 poena)

U potpunosti implementirati klasu `String` koja predstavlja dinamički niz znakova čiji se sadržaj *ne* može menjati nakon kreiranja. Ova klasa treba da obezbedi sledeće:

- Inicijalizaciju datim nizom znakova tipa `char*`.
- Konstruktor kopije.
- Propisno uništavanje, uz dealokaciju zauzetog prostora.
- Operaciju konkatenacije (spajanja) dva ovakva niza znakova koja vraća novi takav niz.

.

Indeks: _____ Student: _____

4. (10 poena)

Osnovna klasa `Command` grupiše zajedničke osobine svih komandi koje inicira korisnik nekog sistema, a jedna od osnovnih jeste da se one izvršavaju uvek po istom postupku, s tim da se pojedini koraci tog postupka mogu razlikovati za konkretne specifične komande. Zajednički, nepromenljivi postupak izvršavanja komande jeste sledeći:

1. Provera prava izvršavanja te komande od strane tekućeg korisnika koju obavlja statička operacija `Command::checkRights(User*)` (ne treba je realizovati).
2. Provera preduslova izvršavanja komande, što svaka konkretna komanda može da definiše specifično za sebe, ali i ne mora (podrazumevano ne radi ništa).
3. Specifično izvršavanje same konkretne komande.
4. Provera uslova koji moraju da važe nakon završetka izvršavanja komande, što svaka konkretna komanda može da definiše specifično za sebe, ali i ne mora (podrazumevano ne radi ništa).
5. Zapisivanje komande u log fajl koje obavlja statička operacija `Command::log(Command*)` (ne treba je realizovati).

(a) Koji projektni obrazac ovde treba primeniti?

(b) Realizovati sve opisane delove osnovne klase `Command` i izvedenu klasu za jednu specifičnu komandu koja ispisuje neku poruku na standardni izlaz.

(c) Dati primer korišćenja komandi kreiranjem objekata odgovarajuće klase i poziva odgovarajuće operacije.

5. (10 poena)

Potrebno je realizovati uprošćenu simulaciju kretanja posetilaca na koncertu u Beogradskoj areni. Posetioci ulaze u arenu na nekom od više ulaza, provode u sektoru arene koji odgovara tom ulazu određeno vreme i potom odlaze iz arene. Pretpostaviti da zadržavanje u areni traje između T-60 i T+20 minuta (ravnomerna raspodela, T je vreme trajanja koncerta). Po završenoj simulaciji, potrebno je ispisati sledeće rezultate: za svaki sektor ukupni broj posetilaca i najveći broj posetilaca u sektoru u jednom trenutku, kao i ukupan broj posetilaca na koncertu. Pri rešavanju zadatka je dozvoljeno koristiti postojeći programski kod TSS uz eventualne potrebne izmene i dopune.

a) (7 poena) Napisati klase potrebne za opisanu simulaciju. U slučaju upotrebe TSS, navesti eventualne dopune/izmene u postojećim TSS klasama.

b) (3 poena) Napisati programski kod koji stvara i ispravno povezuje odgovarajući broj primeraka potrebnih klasa, te pokreće simulaciju i ispisuje tražene rezultate simulacije. Na kraju simulacije, osloboditi korišćenu dinamičku memoriju.