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 2014.
Datum: 07.02.2014.

Kandidat: _____

Broj Indeksa: _____ *E-mail:*_____

Ispit traje 3 sata.

Nije dozvoljeno korišćenje literature.

Pismeni ispit:		***Rad u toku semestra:***	
Zadatak 1	_____/10	*Projekat*	_____/20
Zadatak 2	_____/10	*Domaći zadaci*	_____/30
Zadatak 3	_____/10		
Zadatak 4	_____/20		
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)

Odgovoriti kratko i precizno na sledeća pitanja.

A) Koju grešku prijavljuje prevodilac u toku prevođenja sledećeg programa? Navesti fazu prevođenja i kojoj se prijavljuje greška.

```
#include <iostream>
using namespace std;
void main () { cout<<f()<<endl; }
int f () { return 1; }
```

B) Dovršiti definiciju klase X tako da važe uslovi navedeni u sledećem listningu programa:

```
X x1(2), x2(3);          // PREVODI SE.
X x3 (x1);               // NE PREVODI SE
cout<<x1+x2<<endl;       // PREVODI SE I ISPISUJE "X(5)".
```

```
#include <iostream>
using namespace std;
class X {
  double val;
  // ...
};
```

C) Šta ispisuje dati program napisan na programskom jeziku C++ i zašto?

```
#include <iostream>
using namespace std;
class X {
public:
 void f1 () { cout<<"X:f1"<<endl;}
};
class Y : public X {
public :
 void f1 () { cout<<"Y:f1"<<endl;}
};
void f (X& x) { x.f1(); }
void main () { f (Y()); }
```

2. (10 poena)

Dat je sledeći program na jeziku C++:

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

class B {
      int v;
public:
      B (int i = 0): v(i) { cout << 1 << v; }
      B (const B& b): v(b.v) { cout << 2 << v; }
      virtual ~B() { cout << -1 << v; }
      virtual bool check (B& b) { return b.v == v; }
};

class D : public B {
public:
      D (int v=5) { cout << 3; }
      virtual ~D() { cout << -2; }
      virtual bool check (B& b) { return !B::check(b); }
};

void main() {
      B* arrb[] = {new D(), new B()};
      arrb[0]->check (D(100));
      for (int i=1; i>=0; i--)
            delete arrb[i];
}
```

Šta ispisuje dati program? Detaljno obrazložiti odgovor.

Odgovor:

3. (10 poena)

Realizovati na programskom jeziku C++ klasu **`BitSet`** koja predstavlja skup bitova neograničenog kapaciteta, pri čemu je potrebno ispuniti sledeće zahteve:

- Objekti klase se prave se na osnovu zadatog kapaciteta, podrazumevano 32. Vrednosti bitova (0 ili 1) moraju biti smešteni u dinamičku memoriju.
- Moguće je napraviti skup bitova kao kopiju drugog skupa bitova i/ili dodeliti jedan skup bitova drugom.
- Predvideti metode za postavljanje i čitanje vrednost bita na zadatoj poziciji. Omogućiti ove funkcionalnosti i preklapanjem operatora za indeksiranje.
- Predvideti metodu za dohvatanje skupa bitova između početne i krajnje pozicije. Vraća se kopija podataka, a stanje objekta ostaje nepromenjeno.
- Implementirati ispis objekta u izlazni tok podataka preklapanjem operatora za ispis.
- Objekti se moraju propisno uništavati.

4. (20 poena)

Posao (*Job*) ima svoj identifikator, vreme potrebno za izvršavanje (*execution time*) i vremenski rok (*deadline*) do kog posao mora da bude završen. Jedinstveni raspoređivač poslova (*Scheduler*) smešta poslove u prioritetni red za čekanje i raspoređuje (uređuje) ih prema definisanoj politici raspoređivanja (*SchedulingPolicy*). Omogućava stavljanje posla u red (*put*), kao i uzimanje najprioritetnijeg posla iz reda (*getNext*). Politika raspoređivanja ima operaciju za utvrđivanje redosleda dva posla (`compare(Job, Job): integer`). Postoje dve konkretne politike raspoređivanja: raspoređivanje prema dužini izvršavanja (*ShortestJobFirstPolicy*), tako što se prioritet daje kraćem poslu, i raspoređivanje prema vremenskim rokovima (*EarliestDeadlineFirstPolicy*) tako što se prioritet daje poslu koji ima kraći (tešnji) vremenski rok. Raspoređivaču je moguće definisati politiku raspoređivanja u toku rada programa. Ako se promeni politika raspoređivanja, a u redu za čekanje postoje poslovi, onda se mora izvrši ponovno raspoređivanje/sortiranje prema novoj politici.

a) Modelovati opisani sistem tako da se postigne njegova maksimalna fleksibilnost u smislu lakoće korišćenja i buduće nadogradnje. Prikazati UML dijagram klasa.

b) Objasniti koji projektni obrasci su uočljivi u postavci zadatka i koje klase učestvuju u obrascu.

c) Implementirati klasu `Scheduler`.