Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Programski Prevodioci 1 (IR4PP1)
Programski Prevodioci (RI4PP)
Nastavnik: Prof. dr Dragan Bojić
Ispitni rok: Januar 2008.
Datum: 09.02.2008.

Napomene:

- *Ispit traje 3 sata. Prvih 30 minuta nije dozvoljeno napuštanje sale. Prvih sat i 30 minuta nije dozvoljeno iznositi ispitne zadatke. Poslednjih 10 minuta ispita nije dozvoljeno napuštanje sale.*

- *<u>Na ispitnoj vežbanci</u> obavezno naznačiti broj osvojenih poena na domaćem zadatku u sledećem formatu:*

  Rok u kome je odbranjen domaći _____/(npr: jun 2007)
  Ukupno poena _____/40

- *Na ispitnoj vežbanci potrebno je naznačiti zadatke koji su radjeni (zaokruživanjem broja zadatka).*

- *Nepoštovanje nekog od pravila povlači negativne poene.*

Pitanja

1. [8] Klasifikacija gramatika po Čomskom.

2. [8] Nacrtati izgled run time steka i navesti sekvencu koda za formiranje pristupne veze procedure D. Procedure C i D su ugneždene u proceduru B, koja je ugneždena u proceduru A. Iz A se poziva B, iz B se poziva C, iz C se poziva D.

Zadaci

1. [8]

   a. [3] Pronaći sva suvišna stanja u automatu prikazanom na slici. Rešenje prikazati po koracima. Dati obrazloženje za sprovodjenje navedenih koraka.

	0	1	2	
S	R	R	P	0
D	P	R	S	0
P	A	V	K	1
A	R	S	S	0
R	R	R	P	0
C	D	S	A	0
K	K	K	K	1
E	C	D	R	0
V	P	A	K	0

   b. [5] Neka su konačni automat, stanje konačnog automata i ulazni simbol opisani sledećim klasama respektivno (na jeziku Java):

```
public class FA {
    // vraca sve ulazne simbole konacnog automata.
    public Symbol[] getSymbols() { … }

    // vraca sva stanja konacnog automata.
    // Stanje sa indeksom 0 jeste startno stanje automata.
    public State[] getStates() { … }

    //Vraca stanje u koje se prelazi ukoliko je state
    //trenutno stanje a symbol simbol na ulazu.
    public State nextState(State state, Symbol symbol) { … }
}
```

```
public class State {
  …
}
```

```
public class Symbol {
    …
}
```

Pored navedenih struktura pretpostaviti postojanje (sledeće) klase za opis kolekcije:

```
public class Collection {
    // dodaje stanje u kolekciju. Ne proverava dupilikate!
    public void add(State state) { … }
    // dohvata stanje sa pozicije index.
    public State get(int index) { … }
    // vraca broj stanja u kolekciji
    public int size() { … }
    // da li kolekcija sadrzi navedeno stanje ili ne
    public boolean contains(State state) { … }
}
```

Potrebno je napisati statičku metodu (na jeziku Java, upotrebnom samo navedenih klasa) čija je deklaracija:

```
public static void yq(FA fa, Collection rch, Collection urch);
```

Metoda treba da odredi skupove dostižnih (rch) i suvišnih (urch) stanja. Smatrati da su prosledjene kolekcije prazne. Ne treba vršiti proveru fa, rch i urch na null. Broj stanja i ulaznih simbola automata je sigurno veći od 0. Detaljno komentarisati kod!

2. [4] Pronaći bezkontekstne gramatike koje generišu sledeće:

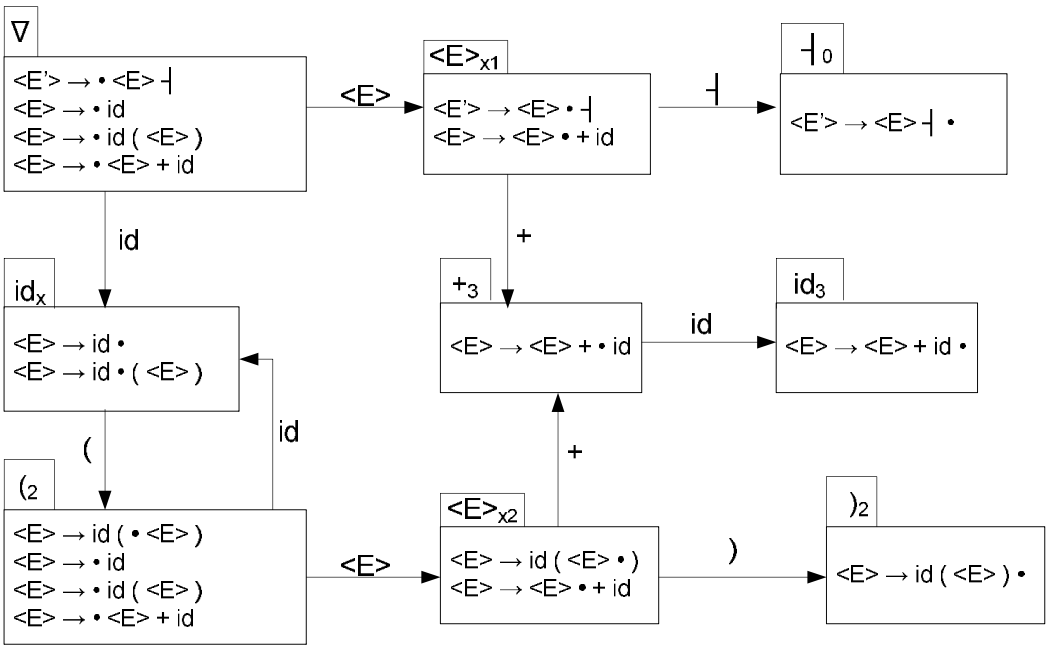
   a. [2] Palindrome za azbuku {a, b}. (Stringovi koji su isti bilo da se čitaju sa leva udesno ili obrnuto). Objasniti predloženo rešenje.

   b. [2] $\{1^n 0^{1+2n}\}$, n >= 1 . Objasniti predloženo rešenje.

3. [10] Data je sledeća gramatika sa startnim simbolom <E>:

   1. <E> → id
   2. <E> → id (<E>)
   3. <E> → <E> + id

   Na osnovu date gramatike kreiran je sledeći karakteristični automat (detektor ručki):



Takodje odredjeni su FIRST i FOLLOW skupovi proširene gramatike:

FIRST(<E'>) = {id}
FIRST(<E>) = {id}
FOLLOW(<E'>) = { -| }
FOLLOW(<E>) = { -|) + }

a. [4] Da li je gramatika LR(0)? Nacrtati potisnu i kontrolnu tabelu.

b. [3] Da li je gramatika SLR(1)? Nacrtati potisnu i kontrolnu tabelu.

c. [3] Prikazati rad parsera iz prethodne tačke, po koracima (prvih 5 koraka), pri parsiranju ulaza "id(id + id)". U svakom koraku prikazati izgled steka, preostali ulaz i akciju koja se izvršava.

4. [12] Za sledeću gramatiku sa startnim simbolom <A>:

1. <A> → <B> <C>
2. <C> → + <B> <C>
3. <C> → ε
4. <B> → <D> <E>
5. <E> → * <D> <E>
6. <E> → ε
7. <D> → (<A>)
8. <D> → a

a. [3] Odrediti FOLLOW skupove za svaki neterminal.

b. [3] Odrediti SELECT skupove za svaku smenu.

c. [6] Konstruisati parser na principu rekurzivnog spusta za datu gramatiku.

5. [10] Za programski fragment:

```
for (i = min, j = max; i < len; i++, j--) {
  e = *p;
  b[j] = e;
  if (i == j) {
    aa = a[i];
    bb = b[j];
  }
  e = 0;
}
aa++;
```

a. [4] Napisati troadresni kod.

b. [3] Napisati implementaciju koda dobijenog u tački a. putem četvorki (prvih šest koraka).

c. [3] Nacrtati graf toka kontrole na nivou osnovnih (bazičnih) blokova.