Elektrotehnički fakultet u Beogradu

Predmet: Programski prevodioci 1
Nastavnik: doc. dr Dragan Bojić
Ispitni rok: Januar 2011.
Datum: 13.01.2011.

Kandidat: _____

Broj Indeksa: _____ *E-mail:*_____

Ispit traje 3 sata. Nije dozvoljeno korišćenje literature.

Zadatak 1	_____/6	*Zadatak 5*	_____/7
Zadatak 2	_____/7	*Zadatak 6*	_____/7
Zadatak 3	_____/9	*Zadatak 7*	_____/10
Zadatak 4	_____/6	*Zadatak 8*	_____/8

Domaći zadatak: _____/40

Ocena: _____

Ukupno: _____/60

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. Na pitanja odgovarati **čitko i precizno**. Srećno!

1) **(6 poena)**

a) Šta je to sistem tipova?

b) Šta je provera tipa i kakve vrste provere postoje?

c) Kada se jezik naziva strogo tipiziranim?

Odgovor:

1. a) Sistem tipova je skup pravila kako se dodeljuju tipovi različitim delovima programa (deklaracijama, izrazima, iskazima).

b) Proverom tipa implementira se određeni sistem tipova. Provera može biti:
- statička, ako je vrši kompajler, ili
- dinamička, ako se izvodi u vreme izvršavanja programa

c) Jezik se naziva strogo tipiziranim ako uspešna statička provera tipova garantuje da u izvršavanju programa neće doći do greške tipa.

2) **(7 poena)**

Posmatra se sledeća situacija (radi se o poslednjoj naredbi u bazičnom bloku):

Međukod	Život i sledeće korišć.	Deskriptori registara	Adresni deskriptori
a = t - u	a(ž,-) t(m,-) u(m,-)	EAX:t EBX:u	a:mem t:EAX u:(mem, EBX)

a) Objasniti po koracima primenu jednostavnog algoritma generisanja koda za dati primer.

b) Prikazati generisani mašinski kod i sadržaj svih deskriptora posle toga.

Rešenje:

Korak 1. Poziva se getreg(t), koji vraća EAX i uklanja EAX iz deskriptora za t.
Korak 2. Generiše se SUB EAX, EBX
Korak 3. Ažuriraju se deskriptori: EAX:a EBX:u a:EAX t:- u:(mem, EBX)
Korak 4. Pošto je kraj bloka, a je živo i nalazi se u registru, generiše se MOV a', EAX

3) **(9 poena)**

Validna email adresa se sastoji od sledećih delova (navode se prema redosledu formiranja adrese):

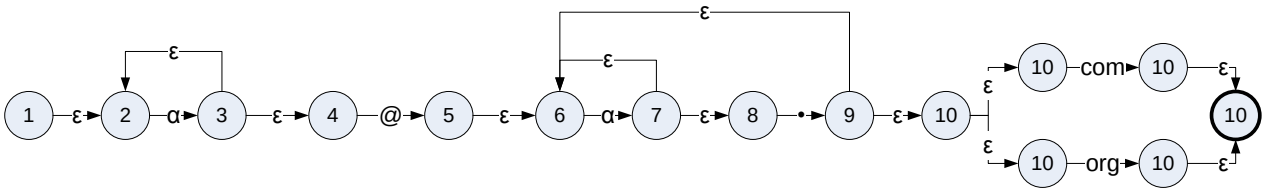
- korisničko ime (niz alfanumeričkih karaktera),
- znak ”@”,
- naziv domena (niz alfanumeričkih karaktera) nakon kojeg sledi ili “.com” ili “.org”. Mogu da postoje i poddomeni, čiji se nazivi odvajaju tačkom. Npr. neke validne adrese su: johnsmith@domain.com, johnsmith@domain1.domain2.org

a) Napisati regularni izraz kojim se opisuju validne mejl adrese.
b) Primenom Tompsonovog algoritma, na osnovu regularnog izraza konstruisati NKA.
c) Prevesti NKA u minimizovani DKA. DKA prikazati u vidu grafa stanja i prelaza.

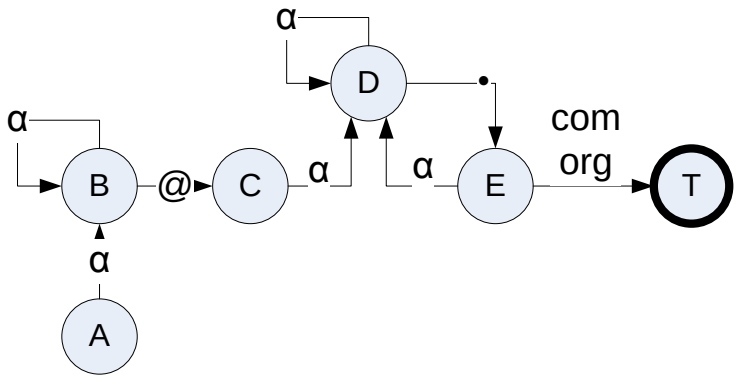
Rešenje:

Regularni izraz:
alphanum = [a-zA-Z0-9]
alphanum+@(alphanum+\.)+(org | com)

NKA:



DKA:



(6 poena)

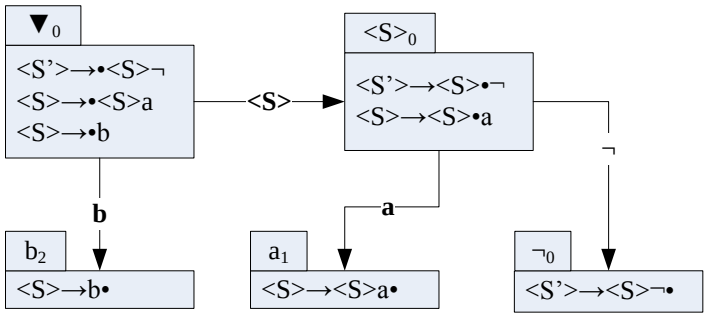
Data je sledeća gramatika:

1. <S> → <S>a
2. <S> → b

a) Da li je data gramatika LR(0)?
b) Konstruisati SLR(1) parser koji parsira sve sekvence opisane datom gramatikom.

Rešenje:

Karakteristični LR(0) automat. Data gramatika jeste LR(0).
Follow(<S>) = {a, ¬}



Potisna tabela SLR(1) automata.				
	<S>	a	b	¬
▼0	<S>0		b2	
<S>0		a1		¬0
¬0				
a1				
b2				

Kontrolna tabela SLR(1) automata.				
	a	b	¬	
▼0		SHIFT		
<S>0	SHIFT		SHIFT	
¬0				ACCEPT
a1	R(1)		R(1)	
b2	R(2)		R(2)	

4) **(7 poena)**

Sledeću gramatiku, koja opisuje binarne razlomke, dopuniti atributima i akcijama tako da se omogući računanje decimalne vrednosti ulazne sekvence. Npr. decimalna vrednost binarnog razlomka $0.c_1\ c_2\ ..\ c_n$ racuna se po formuli $(c_1 2^{-1} + c_2 2^{-2} + ... + c_n 2^{-n})$. Koristiti isključivo sintetizovane atribute.

1. <F> → 0.<D>
2. <D> → 0<D>
3. <D> → 1<D>
4. <D> → 0
5. <D> → 1

Rešenje:

1. $<F>_{value} \rightarrow 0.<D>_{val}$ { value=val;}
2. $<D>_{val} \rightarrow 0<D>_{v}$ { val=v/2; }
3. $<D>_{val} \rightarrow 1<D>_{v}$ { val=(1+v)/2; }
4. $<D>_{val} \rightarrow 0$ { val=0;}
5. $<D>_{val} \rightarrow 1$ { val=1/2;}

5) **(7 poena)**

Konstruisati parser po principu rekurzivnog spusta za datu gramatiku. Parser treba da radi ispravno za sve moguce sekvence. Ukoliko je potrebno za izradu zadatka, dozvoljeno je sprovoditi i neophodne transformacije nad polaznom gramatikom, pod uslovom da se ne naruši ekvivalencija gramatika u smislu sekvenci koje opisuju!

1. <F> → 0.<D>
2. <D> → 0
3. <D> → 1
4. <D> → 0<D>
5. <D> → 1<D>

Rešenje:

Transformisana gramatika (sprovedena leva faktorizacija):

1. <F> → 0.<D>
2. <D> → 0<Z>
3. <D> → 1<Z>
4. <Z> → <D>
5. <Z> → ε

```
Main {
   In=nextToken();
   F();
   If (in == -|) ACCEPT;
   Else REJECT;
}
```

```
Proc F {
   Case in of
   '0':
     in=nextToken();
     if (in !='.') REJECT;
     in=nextToken();
     D();
   default: REJECT;
}
```

```
Proc D {
   Case in of
   '0','1':
     In=nextToken();
     Z();
   default: REJECT;
}
```

```
Proc Z {
   Case in of
   '0','1': D();
   '-|': return;
}
```

6) **(10 poena)**

Dat je sledeci mikrojava program.
Izgenerisati kompletan bajtkod za funkciju calcScore.

```
class Jan2011
        final int windFactor = 1.2;
        int[]  styleMarks;
        class SkiJump { int distance, score;  }
{
   void calcScore(int[] styleMarks, SkiJump j)
      int styleScore, cnt;
   {
      cnt = styleScore = 0;
      while(styleMarks[cnt] != -1) {
         styleScore += styleMarks[cnt];
         cnt++;
      }
      j.score = styleScore + j.distance;
   }
   void collectMarks(int[] marks)  /*T1*/  {...}
   void main()
      SkiJump j;
   {
      styleMarks = new int[6];
      collectMarks();
      j=new SkiJump;
      j.distance = 100;
      calcScore(styleMarks, j);
   }
}
```

Rešenje:

```
      enter 2,4
      const_0
      const_0
      store_2        //styleScore=0
      store_3        //cnt=0
L1:   const_m1
      load_0         //styleMarks
      load_3         //cnt
      aload          //styleMarks[cnt]
      jeq L2         //styleMarks[cnt]!=-1
      load_0         //styleMarks
      load_3         //cnt
      aload          //styleMarks[cnt]
      load_2         //styleScore
      add
      store_2        //styleScore:= ...
      load_3         //cnt
      const_1        //1
      add            //cnt+1
      store_3        //cnt:=...
      jmp L1
L2:   load_1         //j
      load_2         //styleScore
      load_1         //j
      getfield_0     //j.distance
      add            //j.distance+styleScore
      putfield_1     //j.distance:=...
      exit
      return
```

7) **(8 poena)**

Nacrtati izgleda tabele simbola u trenutku parsiranja označenim sa /*T1*/ u izvornom kodu Mikrojava programa iz zadatka 7. Dovoljno je prikazati samo deo *universe* opsega važenja imena.

Rešenje:

Za crtanje tabele simbola, videti zadatke iz materijala za vežbe.