Elektrotehnički fakultet u Beogradu

Predmet:	Programski prevodioci 1
Nastavnik:	dr Dragan Bojić, red. prof.
Asistenti:	mast.inž. Maja Vukasović mast.inž. Mihajlo Ogrizović
Ispitni rok:	Septembar 2023.
Datum:	16.09.2023.

Potpis dežurnog

Kandidat: _____

Broj Indeksa: _____ *Smer:* _____ *Sala:* _____

Ispit traje 150 minuta.
Nije dozvoljeno korišćenje literature.
Prvih sat vremena nije dozvoljeno napuštati ispit.

Zadatak 1	_____/10	*Zadatak 4*	_____/10
Zadatak 2	_____/10	*Zadatak 5*	_____/10
Zadatak 3	_____/10	*Zadatak 6*	_____/10

Ispit: _____/60 **Ukupno:** _____/100

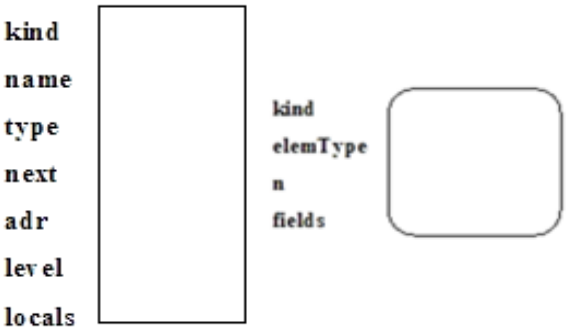
Projekat: _____/40 **Ocena:** _____

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

Podsetnik za neke instrukcije Mikrojava bajtkoda

getstatic	s	... → ..., global[s]
putstatic	s	..., val → ...
new	s	... → ..., adr
getfield	s	..., adr → ..., adr.fields[s]
putfield	s	..., adr, val → ...
const	w	... → ..., w
load	b	... → ..., local[b]
store	b	..., val → ...
new	s	... ..., adr
newarray	b	..., n ..., adr
aload		..., adr, index ..., val
astore		..., adr, index, val ...
baload		..., adr, index ..., val
bastore		..., adr, index, val ...
enter b1, b2		
dup		..., val ..., val, val
dup2		..., v1, v2 ..., v1, v2, v1, v2
dup_x1	..,val2, val1	...,val1, val2, val1
dup_x2	val1, val2, val3	...,val3, val1, val2, val3 ...

Podsetnik strukture čvorova tabele simbola.



1) **(10 poena)**

Dat je deo implementacije klase `Node` u programskom jeziku *Java*, koja predstavlja čvor u sintaksnom stablu izgenerisanom za konstrukciju DKA korišćenjem metoda pozicije, gde su objašnjenja polja i metoda data u komentarima iznad njih (sva polja su inicijalizovana konstruktorom):

```
public class Node {
    private enum Type {
        ASTERISK,   // *
        PLUS,       // +
        PERIOD,     // .
        OR,         // |
        SYMBOL,     // s
        EPSILON     // ε
    }
    // left subtree of position node, null if leaf (symbol) node
    private Node left;

    // right subtree of position node, null if unary operation or leaf (symbol) node
    private Node right;

    // type of position node
    private Type type;

    // position of symbol in regex, only valid for leaf (symbol) nodes, otherwise set
    // to -1
    private int symbolPos;

    // check whether the subsequence with the given node as root of the subtree can be
    // empty
    public boolean isNullable();

    // find set of leaf (symbol) node positions which could be at the beginning of
    // subsequence with the given node as root of the subtree
    public Set<Integer> firstPos();

    // find set of leaf (symbol) node positions which could be at the end of
    // subsequence with the given node as root of the subtree
    public Set<Integer> lastPos();

    // returns a union of the two parameters
    public static Set<Integer> union(Set<Integer> first, Set<Integer> second);

}
```

Potrebno je napisati implementaciju metoda `firstPos()` i `lastPos()`.

Rešenje:

2) **(10 poena)**

Za rekurzivni spust koji opisuje navedenu gramatiku, koji su opisani ispod:

a) Dopuniti delove rekurzivnog spusta obeleženih sa labelama PN (N označava labelu treba popuniti i može predstavljati jednu ili više linija koda), ukoliko je poznato da se sekvence *aadddbb*, *dd* i *ab* prihvataju.

b) Napisati LL(1) gramatiku opisanu rekurzivnim spustom.

```
    Glavni program:
    IN = NEXTCHAR();
    call PROCS;
    if (INP<> '┤' )
        then    REJECT;
        else    ACCEPT;
    end if;
    end;

    procedure PROCS:
        case IN of
            'a':  goto P1;
            'd', 'b', '┤': goto P2;
            default: REJECT;
        end case;
P1:     /* ... */
P2:     /* ... */
    end procedure;
```

```
procedure PROCA:
        case IN of
            'b', '┤': goto P3;
             'd' : goto P4;
            default: REJECT;
        end case;
P3:     /* ... */
P4:     /* .. */
    end procedure;
```

Rešenje:

Odgovor za stavku a):

P1:	P2:	P3:	P4:

3) **(10 poena)**

Za dati Mikrojava bajtkod:

a) Navesti izlaz programa.

b) Rekonstruisati izvorni Mikrojava kod na osnovu datog bajtkoda.

Rešenje:

```
0: enter 2 2
3: load_0
4: load_1
5: mul
6: jmp 3 (=9)
9: exit
10: return
11: enter 2 3
14: const_0
15: store_2
16: load_2
17: load_1
18: jge 21 (=39)
21: load_0
22: load_2
23: aload
24: const_0
25: print
26: const 32
31: const_0
32: bprint
33: inc 2,1
36: jmp -20 (=16)
39: const 10
44: const_0
45: bprint
46: exit
47: return
48: enter 0 5
51: const_0
52: store_0
53: const 10
58: newarray 1
60: store_1
61: const 10
66: newarray 1
68: store_2
69: load_0
70: const 10
75: jge 74 (=149)
78: load_1
79: load_0
80: load_0
81: const 10
86: mul
87: load_0
88: const_3
89: rem
90: add
91: astore
92: load_0
93: const_2
94: rem
95: const_1
96: jne 12 (=108)
99: load_1
100: load_0
101: const_1
102: neg
103: load_1
104: load_0
105: aload
106: mul
107: astore
108: load_1
109: load_0
110: aload
111: const_2
112: call -112 (=0)
115: store_3
116: load_1
117: load_0
118: aload
119: const_3
120: call -120 (=0)
123: store 4
125: load_3
126: load 4
128: jle 10 (=138)
131: load_2
132: load_0
133: load_3
134: astore
135: jmp 8 (=143)
138: load_2
139: load_0
140: load 4
142: astore
143: inc 0,1
146: jmp -77 (=69)
149: load_1
150: const 10
155: call -144
(=11)
158: load_2
159: const 10
164: call -153
(=11)
167: exit
168: return
```

4) **(10 poena)**

Za datu gramatiku konstruisati:
a) LR(0) karakteristični automat
b) LALR(1) karakteristični automat.
Oba automata prikazati u vidu grafa prelaza, unutar stanja napisati konfiguracije.

c) Za oba slučaja LR(0) i LALR(1) navesti stanja sa po jednim konfliktom i objasniti o kom se konfliktu radi, u slučaju da konflikt postoji. Nije potrebno konstruisati tabele parsera.

1. <S> → (<X>)
2. <S> → [<X>]
3. <S> → (<Y>]
4. <S> → [<Y>)
5. <X> → a
6. <Y> → a

Rešenje:

5) **(10 poena)**

Nacrtati graf memorijske predstave objekata c, d1 i d2

```
class A { int x; }
class B { int y; }
class C extends A, B { int z; }
class D extends B,C { int v; }
C c; D d1, d2;
```

a) Ako je dozvoljen neiskorišćen prostor na nivou objekta.

b) Ako neiskorišćen prostor može da postoji samo na nivou klase, a ne i objekta.

Rešenje:

6) **(10 poena)**

Dat je sledeći program na jeziku sličnom Pascalu. Statičko okruženje za nelokalne promenljive je realizovano pomoću pristupnih veza. Glavni program poseduje sopstveni aktivacioni zapis na steku.

a) Prikazati jasno i precizno izgled steka poziva neposredno pre povratka iz procedure c. Voditi računa o formatu aktivacionih zapisa.

b) Napisati kompletan 80x86 asemblerski kod koji bi kompajler izgenerisao za procedure c i d (enter i leave instrukcije ne postoje).

```
program Sep23 (output);
      var
 g, t: integer;

      procedure b(p: integer)
            var m: integer;

            procedure c ()
            begin
                  g:=m+t;
            end; {c}
            procedure d (p:integer)
                  t:=p+2;
                  c();
            end {d}
      begin
            m:=p+1;
            d(m);
      end; {b}

begin
      g:= 1;
      b(g);
end. {Sep23}
```

Rešenje:

Dodatni prostor za rad: