1. Provjeriti da li je gramatika zadata sledecim skupom smena LL1 gramatika.
    1. S→aABbC
    2. S →ε
    3. A→ASB
    4. A→ε
    5. B→Sac
    6. B→Ac
    7. B→ε
    8. C→Sd
    9. C→Ca

   Ovo bi mogla biti LL1 gramatika sa ε pravilima. Neophodno je provjeriti. Uslov koji treba ispuniti je:

   $FIRST(\alpha_i \circ FOLLOW(A)) \cap FIRST(\alpha_j \circ FOLLOW(A)) = \Phi,\ i \neq j$

   $FIRST(aABbC \circ FOLLOW(S)) \cap FIRST(\varepsilon \circ FOLLOW(S)) = \{a\} \cap \{a, d\} = \{a\}$

   $FIRST(ASB \circ FOLLOW(A)) \cap FIRST(\varepsilon \circ FOLLOW(A)) = \{a, c\} \cap \{a, c\} = \{a, c\}$

   $FIRST(Sac \circ FOLLOW(B)) \cap FIRST(Ac \circ FOLLOW(B)) = \{a\} \cap \{a, c\} = \{a\}$
   $FIRST(Sac \circ FOLLOW(B)) \cap FIRST(\varepsilon \circ FOLLOW(B)) = \{a\} \cap \{b, a, c\} = \{a\}$
   $FIRST(Ac \circ FOLLOW(B)) \cap FIRST(\varepsilon \circ FOLLOW(B)) = \{c, a\} \cap \{b, a, c\} = \{a, c\}$

   $FIRST(Sd \circ FOLLOW(C)) \cap FIRST(Ca \circ FOLLOW(C)) = \{a, d\} \cap \{a\} = \{a\}$

   Ova gramatika ne zadovoljava uslov LL1 gramatika sa ε pravilima.

2. Kreirati YACC (i odgovarajuci LEX) specifikaciju za generisanje sintaksnog analizatora jezika zadtaog sledecim skupom smjena:

   Program→**START** NizNaredbi **END**
   NizNaredbi→NizNaredbi;Naredba | Naredba
   Naredba→IfNaredba |Dodela
   Dodela→**ID**=Izraz
   Izraz→**ID | CONST**
   IfNaredba→**IF** Izraz **THEN** Naredba ElseDeo
   ElseDeo→**ELSE** Naredba |ε

   Token **CONST** označava cjelobrojnu konstantu čiji je zapis definisan sledecim sablonom:

   [0[X]]<niz_cifara_zadate_osnove>

   Pri čemu je osnova 16,ukoliko broj pocinje nizom 0X;8 ako broj počinje nulom;10 u ostalim slučajevima. Token **ID** označava identifikator (niz slova I cifara u kojem prvi

znak ne moze da bude cifra). Generisani sintaksni analizator treba da sadrzi I oporavak od mogucih gresaka u kodu.

/zadatak.l

```
%{
    #include <stdlib.h>
    #include "zadatak.tab.h"
    void yyerror(char *poruka);
    int br_linija = 1;
%}
%%
START { ECHO; return START;}
END { ECHO; return END;}
IF { ECHO; return IF;}
THEN { ECHO; return THEN;}
ELSE { ECHO; return ELSE;}
([a-zA-Z_])([0-9a-zA-Z_]*) {ECHO; return ID;}
(([0]?X)?)[0-9]+ {ECHO; return CONST;}
[;=] {ECHO; return *yytext;}
[ \t] {;}
[\n] { br_linija++; ECHO; printf(" %d ", br_linija);}
%%
int yywrap() {
  return 1;
}
```

/zadatak.y

```
%token ID CONST START END IF THEN ELSE
%{
  #include <stdio.h>
  extern int yylex();
  void yyerror(char *poruka);
  extern int br_linija;
  extern FILE *yyin;
  int br_gresaka = 0;
  int yydebug=1;
  int flag=0;
%}
%%
Start:
        Program '\n' {;}
        |
        ;
Program:
        START NizNaredbi END {;}
        | START NizNaredbi error
                        {printf("\nnr 1 Doslo je do greske na
%d. liniji, ocekivano ';'\n", br_linija); br_gresaka++;}
        | START error
```

```
                        { if (flag!=br_linija) {
                              printf("\nnr 2 Doslo je do greske
na %d. liniji, ocekivano '}'\n", br_linija);
                              br_gresaka++;
                              flag=br_linija;
                              }
                              }
        ;
NizNaredbi:
        NizNaredbi ';' Naredba  {;}
        | NizNaredbi ';' error
                {if (flag==br_linija) {
                  printf("\nnr 3 Doslo je do greske na %d.
liniji.\n", br_linija);
                  br_gresaka++;
                  flag=br_linija;
                  }
                  }
         | NizNaredbi error
                {if (flag==br_linija) {
                  printf("\nnr 4 Doslo je do greske na %d.
liniji.\n", br_linija);
                  br_gresaka++;
                  flag=br_linija;
                  }
                  }

        | Naredba  {;}
        ;
Naredba:
        IfNaredba {;}
        | Dodela {;}
        ;
Dodela:
        ID '=' Izraz {;}
        | ID '=' error {if (flag!=br_linija) {
                              printf("\nnr 5 Greska na
%d. liniji.\n", br_linija);
                              br_gresaka++;
                              flag=br_linija;
                              }
                              }
        | ID error { if (flag!=br_linija) {
                              printf("\nnr 6 Greska na %d.
liniji, fali '='.\n", br_linija);
                              br_gresaka++;
                              flag=br_linija;
                              }
                              }
        ;
Izraz:
   ID {;}
        | CONST {;}
        ;
IfNaredba:
        IF Izraz THEN Naredba ElseDeo {;}
```

```
        | IF Izraz THEN Naredba {;}
        | IF Izraz THEN Naredba error {if (flag!=br_linija) {
                                                printf("\nnr 7
Greska na %d. liniji\n", br_linija);
                                                br_gresaka++;
                                                flag=br_linija;
                                                }
                                                }

        | IF Izraz THEN error {if (flag!=br_linija) {
                                                printf("\nnr 8
Greska na %d. liniji\n", br_linija);
                                                br_gresaka++;
                                                flag=br_linija;
                                                }
                                                }

        | IF Izraz error {if (flag!=br_linija) {
                                                printf("\nnr 9
Greska na %d. liniji\n", br_linija);
                                                br_gresaka++;
                                                flag=br_linija;
                                                }
                                                }
        | IF error {if (flag!=br_linija) {
                                                printf("\nnr 10
Greska na %d. liniji\n", br_linija);
                                                br_gresaka++;
                                                flag=br_linija;
                                                }
                                                }


        ;
ElseDeo:
        ELSE Naredba {;}
        | ELSE error {if (flag!=br_linija) {
                                                printf("\nnr 11
Greska na %d. liniji\n", br_linija);
                                                br_gresaka++;
                                                flag=br_linija;
                                                }
                                                }

        ;
%%
void yyerror(char *poruka) {
  printf("%s\n", poruka);
}
int main() {
  yyin=fopen("tekst.txt", "r");
  yyparse();
  if (br_gresaka==0) {
     printf("\nIzraz je ispravan.\n");
  } else {
     printf("\nU izrazu se nalazi ukupno %d gresaka.\n",
br_gresaka);
  }
```

```
  fclose(yyin);
  return 0;
}
```

3. Kreirati YACC specifikaciju za generisanje sintaksnog analizatora jezika definisanog sledecom gramatikom:

   Blok→{Opisi;IzvrsneNaredbe}
   Opisi→NOpisa | Opisi;NOpisa
   NOpisa→Type ListaPromjenjivih
   ListaPromjenjivih→ID | ListaPromjenjivih ,ID
   IzvrsneNaredbe::=Naredba | IzvrsneNaredbe ; Naredba
   Naredba→ **ID**=Izraz
   Izraz→Izraz + IzrazT | IzrazT
   IzrazT→IzrazT*IzrazF | IzrazF
   IzrazF→(Izraz) | **CONST | ID**
   Type→**int | float**

   Token CONST ozbacava konstante ciji je zapis definisan sledecim šablonom:

   <niz_cifara>[.[<niz_cifara>]]

   Token ID oznacava identifikator(niz slova,cifara i '_' u kojem prvi znak ne moze da bude cifra).

   /zadatak.l

```
%{
    #include <stdlib.h>
    #include "zadatak.tab.h"
    void yyerror(char *poruka);
    int br_linija = 1;
%}
%%
INT { ECHO; return INT;}
FLOAT { ECHO; return FLOAT;}
([a-zA-Z_])([0-9a-zA-Z_]*) {ECHO; return ID;}
[0-9]+\.[0-9]+ {ECHO; return CONST;}
[;=+*\{\}\(\)\,] {ECHO; return *yytext;}
[ \t] {;}
[\n] { br_linija++; ECHO; printf(" %d ", br_linija);}
%%
int yywrap() {
  return 1;
}
```

   /zadatak.y

```
%token ID CONST INT FLOAT
%left '+'
%left '*'
%{
  #include <stdio.h>
  extern int yylex();
```

```
  void yyerror(char *poruka);
  extern int br_linija;
  extern FILE *yyin;
  int br_gresaka = 0;
  int yydebug=1;
  int flag=0;
%}
%%
Start:
        Blok '\n' {;}
        |
        ;
Blok:
        '{' Opisi ';' IzvrsneNaredbe '}' {;}
        | '{' Opisi ';' IzvrsneNaredbe error
                        {printf("\nnr 1 Doslo je do greske na
%d. liniji, ocekivano '}'\n", br_linija); br_gresaka++;}
        | '{' Opisi ';' error
                        { if (flag!=br_linija) {
                              printf("\nnr 2 Doslo je do greske
na %d. liniji.\n", br_linija);
                              br_gresaka++;
                              flag=br_linija;
                              }
                              }
           | '{' Opisi error
                        { if (flag!=br_linija) {
                              printf("\nnr 2 Doslo je do greske
na %d. liniji.\n", br_linija);
                              br_gresaka++;
                              flag=br_linija;
                              }
                              }
| '{' error
                        { if (flag!=br_linija) {
                              printf("\nnr 3 Doslo je do greske
na %d. liniji.\n", br_linija);
                              br_gresaka++;
                              flag=br_linija;
                              }
                              }
        ;
Opisi:
        NOpisa {;}
           | Opisi ';' NOpisa {;}
        | Opisi ';' error
                {if (flag==br_linija) {
                  printf("\nnr 4 Doslo je do greske na %d.
liniji.\n", br_linija);
                  br_gresaka++;
                  flag=br_linija;
                  }
                  }
         | Opisi error
                {if (flag==br_linija) {
```

```
                  printf("\nnr 5 Doslo je do greske na %d.
liniji.\n", br_linija);
                  br_gresaka++;
                  flag=br_linija;
                  }
                  }
        ;
NOpisa:
        Type ListaPromjenljivih {;}
        | Type error
                {if (flag==br_linija) {
                  printf("\nnr 6 Doslo je do greske na %d.
liniji.\n", br_linija);
                  br_gresaka++;
                  flag=br_linija;
                  }
                  }
        ;
ListaPromjenljivih:
        ID {;}
        | ListaPromjenljivih ',' ID {;}
        | ListaPromjenljivih ',' error { if (flag!=br_linija) {
                                printf("\nnr 7 Greska na %d.
liniji.\n", br_linija);
                                br_gresaka++;
                                flag=br_linija;
                                }
                                }
           | ListaPromjenljivih error { if (flag!=br_linija) {
                                printf("\nnr 8 Greska na %d.
liniji.\n", br_linija);
                                br_gresaka++;
                                flag=br_linija;
                                }
                                }
        ;
IzvrsneNaredbe:
        Naredba {;}
        | IzvrsneNaredbe ';' Naredba {;}
        | IzvrsneNaredbe ';' error { if (flag!=br_linija) {
                                printf("\nnr 9 Greska na %d.
liniji.\n", br_linija);
                                br_gresaka++;
                                flag=br_linija;
                                }
                                }
           | IzvrsneNaredbe error { if (flag!=br_linija) {
                                printf("\nnr 10 Greska na %d.
liniji.\n", br_linija);
                                br_gresaka++;
                                flag=br_linija;
                                }
                                }
        ;
Naredba:
   ID '=' Izraz {;}
```

```
        | ID '=' error { if (flag!=br_linija) {
                                printf("\nnr 11 Greska na %d.
liniji.\n", br_linija);
                                br_gresaka++;
                                flag=br_linija;
                                }
                                }
          | ID error { if (flag!=br_linija) {
                                printf("\nnr 12 Greska na %d.
liniji.\n", br_linija);
                                br_gresaka++;
                                flag=br_linija;
                                }
                                }

        ;
Izraz:
        Izraz '+' IzrazT {;}
     | Izraz '+' error { if (flag!=br_linija) {
                                printf("\nnr 13 Greska na %d.
liniji.\n", br_linija);
                                br_gresaka++;
                                flag=br_linija;
                                }
                                }
     | Izraz error { if (flag!=br_linija) {
                                printf("\nnr 14 Greska na %d.
liniji.\n", br_linija);
                                br_gresaka++;
                                flag=br_linija;
                                }
                                }
     | IzrazT {;}
        ;
IzrazT:
        IzrazT '*' IzrazF {;}
     | IzrazT '*' error { if (flag!=br_linija) {
                                printf("\nnr 15 Greska na %d.
liniji.\n", br_linija);
                                br_gresaka++;
                                flag=br_linija;
                                }
                                }
     | IzrazT error { if (flag!=br_linija) {
                                printf("\nnr 16 Greska na %d.
liniji.\n", br_linija);
                                br_gresaka++;
                                flag=br_linija;
                                }
                                }
     | IzrazF {;}
        ;
IzrazF:
        '(' Izraz ')' {;}
     | '(' error { if (flag!=br_linija) {
```

```
                                printf("\nnr 17 Greska na %d.
liniji.\n", br_linija);
                                br_gresaka++;
                                flag=br_linija;
                                }
                                }
      | CONST {;}
      | ID {;}
        ;
Type:
        INT {;}
        | FLOAT {;}
        ;
%%
void yyerror(char *poruka) {
  printf("%s\n", poruka);
}
int main() {
  yyin=fopen("tekst.txt", "r");
  yyparse();
  if (br_gresaka==0) {
     printf("\nIzraz je ispravan.\n");
  } else {
     printf("\nU izrazu se nalazi ukupno %d gresaka.\n",
br_gresaka);
  }
  fclose(yyin);
  return 0;
}
```

4. Kreirati LR sintaksnu tebelu za gramatiku zadatu sledecim skupom smjena:
    1. CaseList→CaseListCaseItem
    2. CaseList→CaseItem
    3. CaseItem→case CONST:Assignment break;
    4. Assignment→ID=Expression;
    5. Expression→ID
    6. Expression→CONST

   Korišćenjem kreirane tabele provjeriti da li je jezik koji je definisan ovom gramatikom pripada zapis:

   case CONST: ID=CONST;
   case CONST:ID=ID;break;

I_0
CaseList' → · CaseList
CaseList → · CaseList CaseItem
CaseList → · CaseItem
CaseItem → · **case CONST :** Assignment **break ;**

I_1=goto(I_0,CaseList)
CaseList’ → CaseList ·
CaseList → CaseList · CaseItem
CaseItem → · **case CONST :** Assignment **break ;**

I_2=goto(I_0,CaseItem)
CaseList → CaseItem ·

I_3=goto(I_0,**case**)
CaseItem → **case** · **CONST :** Assignment **break ;**

I_4=goto(I_1,CaseItem)
CaseList → CaseList CaseItem ·

~~I_5~~=goto(I_1,**case**) = I_3

I_5=goto(I_3,**CONST**)
CaseItem → **case CONST** · **:** Assignment **break ;**

I_6=goto(I_5,**:**)
CaseItem→ **case CONST :** · Assignment **break ;**
Assignment → · **ID =** Expression **;**

I_7=goto(I_6,Assignment)
CaseItem → **case CONST :** Assignment · **break ;**

I_8=goto(I_6,**ID**)
Assignment → **ID** · **=** Expression **;**

I_9=goto(I_7,**break**)
CaseItem → **case CONST :** Assignment **break** · **;**

I_{10}=goto(I_8,**=**)
Assignment → **ID =** · Expression **;**
Expression → · **ID**
Expression → · **CONST**

I_{11}=goto(I_9,**;**)
CaseItem → **case CONST :** Assignment **break ;** ·

I_{12}=goto(I_{10},Expression)
Assignment → **ID =** Expression · **;**

I_{13}=goto(I_{10},**ID**)
Expression → **ID** ·

I_{14}=goto(I_{10},**CONST**)
Expression → **CONST** ·

I_{15}=goto(I_{12},**;**)
Assignment → **ID =** Expression **;** ·

Stanje	Akcije								Prelazi			
	case	CONST	:	break	;	ID	=	#	CaseList	CaseItem	Assignment	Expression
0	s3								1	2		

1	s3							acc		4		
2	r2							r2				
3		s5										
4	r1							r1				
5			s6									
6						s8					7	
7				s9								
8							s10					
9					s11							
10		s14				s13						12
11	r3							r3				
12					s15							
13					r5							
14					r6							
15				r4								

FOLLOW(CaseList)={case, #}
FOLLOW(CaseItem)={case, #}
FOLLOW(Assignment)={break}
FOLLOW(Expression)={;}

Niz u magacinu	Ulazni niz	Akcija automata
0	case CONST: ID=CONST; case CONST:ID=ID;break; #	Shift 3
0 case 3	CONST: ID=CONST; case CONST:ID=ID;break; #	Shift 5
0 case 3 CONST 5	: ID=CONST; case CONST:ID=ID;break; #	Shift 6
0 case 3 CONST 5 : 6	ID=CONST; case CONST:ID=ID;break; #	Shift 8
0 case 3 CONST 5 : 6 ID 8	=CONST; case CONST:ID=ID;break; #	Shift 10
0 case 3 CONST 5 : 6 ID 8 = 10	CONST; case CONST:ID=ID;break; #	Shift 14
0 case 3 CONST 5 : 6 ID 8 = 10 CONST 14	; case CONST:ID=ID;break; #	Reduce Expression→CONST
0 case 3 CONST 5 : 6 ID 8 = 10 Expression 12	; case CONST:ID=ID;break; #	Shift 15
0 case 3 CONST 5 : 6 ID 8 = 10 Expression 12 ; 15	case CONST:ID=ID;break; #	Error!