Tema Lex & Yacc: Verificator de expresii aritmetice si logice

1. Descriere generala

Se doreste implementarea unui program in **Lex** si **Yacc** care sa primeasca de la tastatura sau dintr-un fisier o serie de **declaratii de variabile** si **expresii** care pot fi aritmetice sau logice. Programul trebuie sa **valideze sintactic** aceste expresii si sa **evalueze rezultatul** lor daca sunt corecte.

2. Exemplu de input valid

```
int a = 5;
int b = 2;
bool ok = true;
a + b * 3
(a + 1) > b && ok
```

3. Comportament asteptat

- Declararea variabilelor: tipul trebuie sa fie `int` sau `bool`.
- Expresiile pot contine: `+`, `-`, `*`, `/`, `()`, `&&`, `||`, `==`, `!=`, `<`, `>`, `<=`, `>=`, precum si variabile deja definite.
- Programul trebuie sa detecteze erori de sintaxa, folosirea de variabile nedeclarate, sau expresii invalide.
- La final, programul va afisa:
  - expresia evaluata (daca este corecta)
  - mesaj de eroare (daca este incorecta)

4. Cerintele proiectului si punctajul

Scorul total este de 10 puncte, din care 1 punct este acordat din oficiu.

1. **Declararea si salvarea variabilelor in tabelul de simboluri (1 punct)**
   Recunoasterea declaratiilor de variabile si salvarea lor in tabelul de simboluri.

2. **Parsarea expresiilor aritmetice** **(1 puncte)**
   Expresiile de forma `a + b * 3` trebuie recunoscute corect, respectand precedenta operatorilor.

3. **Parsarea expresiilor logice** **(1 punct)**
   Expresiile precum `a > b && ok` trebuie analizate corect.

4. **Evaluarea expresiilor (Yacc)** **(2 punct)**
   Calcularea si afisarea rezultatului expresiilor valide, cu verificarea tipurilor.

5. **Tratarea erorilor** **(1 punct)**
   Detectarea variabilelor nedeclarate, tipuri incompatibile, sintaxa gresita.

6. **Definirea corecta a gramaticii in Yacc** **(1 punct)**
   Reguli clare, recursive, folosind nonterminali si productii compuse.

7. **Definirea corecta a tokenilor in Lex** **(1 punct)**
   Recunoasterea tokenilor: identificatori, constante, operatori, delimitatori.

8. **Cod structurat si comentat** **(1 punct)**
   Cod lizibil, organizat, cu comentarii relevante.

9. **Punct din oficiu** **(1 punct)**
   Acordat automat.

5. Cerinte tehnice

- Folositi `Lex` pentru tokenizare.
- Folositi `Yacc` pentru analiza sintactica si semantica.
- Tabelul de simboluri poate fi implementat manual (vector sau hashmap).
- Input-ul se poate citi de la tastatura sau dintr-un fisier redirectat.

6. Sugestii de implementare

- In Lex, definirea tokenilor: `INT`, `BOOL`, `ID`, `NUM`, `TRUE`, `FALSE`, `AND`, `OR`, `EQ`, `NE`, etc.
- In Yacc, reguli pentru expresii:

```
expr: expr '+' expr
    | expr '*' expr
    | '(' expr ')'
    | ID
    | NUM
    ;
```