LABORATOR, PROIECTUL 2 (fiecare student va alege un singur proiect)

DEADLINE: SĂPTĂMÂNA 9-16 IANUARIE 2026

ÎN SĂPTĂMÂNILE ANTERIOARE DEADLINE-ULUI PUTEȚI VENI LA ORICE LABORATOR PENTRU A PUNE ÎNTREBĂRI SAU PENTRU A PREDA PROIECTUL

MAXIM 2 STUDENȚI DIN ACEEAȘI GRUPĂ POT ALEGE UNA DINTRE TEMELE 1-14, DAR ACEASTA VA FI FĂCUTĂ INDIVIDUAL. TEMELE LA CARE SE POATE FACE ECHIPĂ DE 2 PERSOANE VOR FI ALESE DE O SINGURĂ ECHIPĂ

FIECARE GRUPĂ VA CREA UN FIȘIER ÎN FILES CU NUMELE 36x_tema_aleasa pe care îl va completa fiecare student din grupa respectivă $(\mathbf{x} \in (\mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}\})$**:**

NUME ȘI PRENUME nr-temei

Veți încărca proiectul în assignment-ul PROIECT 2 sub forma unui fișier text pe care îl veți verifica cu Turnitin

1. Să se realizez un program care simulează funcționarea unui automat finit nedeterminist cu λ-tranziții. Programul citește dintr-un fișier elementele unui automat finit nedeterminist cu λ - tranzitii oarecare (stările, starea inițială, stările finale, alfabetul automatului, tranzițiile). Programul permite citirea unui număr oarecare de șiruri peste alfabetul de intrarea al automatului. Pentru fiecare astfel de șir se returnează DA sau NU, după cum șirul respectiv aparține sau nu limbajului acceptat de automat.

2. Program care simulează funcționarea unui translator finit nedeterminist cu λ-tranziții. Programul citește dintr-un fișier elementele unui translator finit nedeterminist cu λ-tranziții oarecare (stările, starea inițială, stările finale, alfabetul de intrare, alfabetul de ieșire, tranzițiile). Programul permite citirea unui număr oarecare de șiruri peste alfabetul de intrarea al translatorului. Pentru fiecare astfel de șir se afișează toate ieșirile (șiruri peste alfabetul de ieșire) corespunzătoare (Atenție! pot exista 0, 1 sau mai multe ieșiri pentru același șir de intrare).

3. Să se scrie un program care primește la intrare elementele unei expresii regulate (alfabetul expresiei, expresia propriu-zisă (în forma infixată - adică forma naturală), care conține 3 tipuri de operatori: reuniune (|), concatenare (.) și iterație Kleene (*)).

a) Să se determine forma postfixată a expresiei cu algoritmul Shunting Yard.

b) Folosind forma postfixată a expresiei, să se determine un automat finit nedeterminist cu λ-tranziții aplicând algoritmul Thompson, care recunoaște același limbaj ca cel descris de expresia regulată.

c) Programul afisează și graful care corespunde noului automat.

(pot face echipă 2 persoane)

4. Să se scrie un program care primește la intrare elementele unei expresii regulate (alfabetul expresiei, expresia propriu-zisă (în forma infixată - adică forma naturală), care conține 3 tipuri de operatori: reuniune (|), concatenare (.) și iterație Kleene (*)).

a) Să se determine forma postfixată a expresiei cu algoritmul Shunting Yard și să se obțină arborele sintactic ce corespunde expresiei.

b) Pe baza arborelui sintactic, se vor construi mulțimile $firstpos, lastpos, nullable, followpos$, precum și automatul finit determinist care recunoaște același limbaj ca cel descris de expresia regulată.

c) Programul afișează și graful care corespunde noului automat.

(pot face echipă 2 persoane)

5. Să se realizeze un program care simulează functionarea unui automat push-down nedeterminist cu λ-tranziții și stări finale. Programul citește elementele unui automat push-down nedeterminist cu λ-tranzitii oarecare (stările, starea inițială, stările finale, alfabetul automatului, alfabetul stivei, simbolul inițial al stivei, tranzițiile). Programul permite citirea unui număr oarecare de șiruri peste alfabetul automatului. Pentru fiecare astfel de șir se afișează 'DA' sau 'NU', după cum șirul este sau nu este acceptat de automat.

Observație. Puteți implementa automatul cu vidarea stivei.

6. Să se realizeze un program care simulează funcționarea unui translator stivă nedeterminist cu λ-tranziții și stări finale. Programul citește elementele unui translator stivă nedeterminist cu λ-tranziții oarecare (stările, starea inițială, stările finale, alfabetul de intrare, alfabetul de ieșire, alfabetul stivei, simbolul inițial al stivei, tranzițiile). Programul permite citirea unui număr oarecare de șiruri peste alfabetul de intrare al translatorului. Pentru fiecare astfel de șir se afisează toate ieșirile (șiruri peste alfabetul de ieșire) corespunzătoare (Atenție! pot exista 0, 1 sau mai multe ieșiri pentru același șir de intrare).

Observație. Puteți implementa translatorul cu vidarea stivei.

7. Să se implementeze cu ajutorul programului flex (Fast Lexical Analyzer) sau unul dintre variantele sale: jlex sau jflex, un analizor lexical pentru limbajul C, C++ sau Java. Pentru fiecare token recunoscut, se vor returna: șirul din fișierul analizat care corespunde token-ului (lexema), tipul token-ului curent, linia din fișierul de intrare pe care se afla token-ul curent, un mesaj de eroare atunci când este întâlnită o eroare lexicală.

8. Să se implementeze cu ajutorul programului flex (Fast Lexical Analyzer) sau unul dintre variantele sale: jlex sau jflex, un analizor lexical pentru un limbaj la alegere, altul decât C sau C++ sau Java. Pentru fiecare token recunoscut, se vor returna: șirul din fișierul analizat care corespunde token-ului (lexema), tipul token-ului curent, linia din fișierul de intrare pe care se afla token-ul curent, un mesaj de eroare atunci când este întâlnită o eroare lexicală.

9. Să se scrie un program pentru implementarea algoritmului de analiză sintactică Earley. Programul primește la intrare elementele unei gramatici independente de context oarecare, inclusiv cu λ-producții. Programul acceptă un număr oarecare de șiruri peste alfabetul terminalilor. Pentru fiecare șir se creează și se afisează tabelele Earley corespondente pe baza cărora se precizează dacă șirul respectiv este acceptat sau nu. Pentru fiecare astfel de șir se afișează 'DA' sau 'NU' după cum șirul este sau nu este acceptat de automat.

10. Să se scrie un program care implementează algoritmul pentru gramatici LL(1). Programul primește la intrare: elementele unei gramatici independente de context, nerecursivă la stânga, oarecare. Programul determină tabela de analiză sintactică asociată și decide dacă gramatica dată este LL(1). În caz afirmativ, programul permite citirea unui număr oarecare de șiruri peste alfabetul terminalilor. Pentru fiecare șir terminal se determină, pe baza tabelei de analiza sintactică obținute, dacă este în limbajul generat de gramatica respectivă iar in caz afirmativ se afișează derivarea sa stângă (o succesiune de numere, fiecare număr reprezentând numărul producției aplicate) (pot face echipa 2 persoane)

11. Să se scrie un program care implementează algoritmul pentru gramatici SLR(1). Programul primește la intrare elementele unei gramatici independente de context oarecare. Programul determină tabela de analiză sintactică asociată și decide dacă gramatica dată este SLR(1). În caz afirmativ, programul permite citirea unui număr oarecare de șiruri peste alfabetul terminalilor. Pentru fiecare șir terminal se determină, pe baza tabelei de analiza sintactică obținute, dacă este în limbajul generat de gramatica respectivă iar in caz afirmativ se afișează derivarea sa dreaptă (o succesiune de numere, fiecare număr reprezentând numărul producției aplicate) (pot face echipa 2 persoane)

12. Să se scrie un program care implementează algoritmul pentru gramatici LR(1). Programul primește la intrare elementele unei gramatici independente de context oarecare. Programul determină tabela de analiză sintactică asociată și decide dacă gramatica dată este LR(1). În caz afirmativ, programul permite citirea unui număr oarecare de șiruri peste alfabetul terminalilor. Pentru fiecare șir terminal se determină, pe baza tabelei de analiza sintactică obținute, dacă este în limbajul generat de gramatica respectivă iar in caz afirmativ se afișează derivarea sa dreaptă (o succesiune de numere, fiecare număr reprezentând numărul producției aplicate) (pot face echipa 2 persoane)

13. Să se studieze specificația pentru generatorul de parser-e Bison. Să se exemplifice pentru gramatica sintaxei limbajului C++/ Python sau pentru un alt limbaj (se vor considera minim două tipuri de variabile, minim 2 instrucțiuni, dintre care o instrucțiune if și una de ciclare).

14. Să se scrie un program care primește la intrare o gramatică independentă de context G. Pentru fiecare producție $A \rightarrow x$ se va calcula mulțimea $First(x \cdot Follow(A))$. Programul verifică dacă pentru orice 2 producții $A \rightarrow x, A \rightarrow y, x \neq y, First\big(x \cdot Follow(A)\big) \cap First(y \cdot Follow(A)) = \emptyset$. Dacă da, atunci gramatica este LL(1) și se va genera un fișier text care va

conține funcțiile (în C sau C++) corespunzătoare fiecărui neterminal din G, folosind algoritmul recursiv descendent pentru gramatica dată.