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

DEADLINE: 29 APRILIE 2025

PUTEȚI REALIZA PROIECTUL ÎN ORICE LIMBAJ DORIȚI

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

PROIECTUL ESTE INDIVIDUAL

MAXIM 2 STUDENȚI DIN ACEEAȘI SEMIGRUPĂ POT ALEGE ACELAȘI PROIECT. TEMELE LA CARE SE POT FACE ECHIPE DE 2 STUDENȚI POT FI ALESE DE MAXIM 2 ECHIPE

FIECARE STUDENT VA COMPLETA ÎN FIȘIERUL DIN FILES Proiectul ales FPC 2025.docx numărul temei

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

1. Program care simuleaza functionarea unui translator finit nedeterminist cu λ-tranziții. Programul citește (dintr-un fișier sau de la consolă) elementele unui translator finit nedeterminist cu λ-tranziții oarecare (starile, starea initiala, starile finale, alfabetul de intrare, alfabetul de iesire, tranzitiile). 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) corespunzatoare (Atenție! pot exista 0, 1 sau mai multe ieșiri pentru acelasi șir de intrare).

2. Program care simuleaza functionarea unui automat push-down nedeterminist cu λ-tranzitii. Programul citeste elementele unui automat push-down nedeterminist cu λ-tranzitii oarecare (starile, starea initiala, starile finale, alfabetul automatului, alfabetul stivei, simbolul initial al stivei, tranzitiile). Programul permite citirea unui nr oarecare de siruri peste alfabetul automatului. Pentru fiecare astfel de sir se afiseaza `DA` sau `NU` după cum șirul este sau nu este acceptat de automat.

3. Să se scrie un program care primește la intrare elementele unei expresii regulate (alfabetul expresiei și expresia propriu-zisă în forma infixată - adică forma naturală). Expresia poate să conțină paranteze și 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, folosind algoritmul Thompson, care recunoaște

același limbaj ca cel descris de expresia regulată. Programul afișează și graful care corespunde noului automat.

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). Expresia poate să conțină paranteze și 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. Programul afișează și graful care corespunde noului automat. (pot face echipă 2 studenți)

5. Program care simulează funcționarea unui translator stivă nedeterminist cu λ-tranziții. 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, stările finale dacă se lucrează cu stări finale). Programul permite citirea unui număr oarecare de șiruri peste alfabetul de intrare 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).

6. Să se scrie un program pentru implementarea algoritmului de analiză sintactică Earley îmbunătățit. 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 tabelele Earley corespondente pe baza cărora se precizează dacă șirul respectiv este acceptat sau nu (se va afișa mesajul DA sau NU, după caz).

7. Să se scrie un program care implementează algoritmul pentru gramatici $LL(1)$. Programul primește la intrare: elementele unei gramatici independente de context oarecare, nerecursivă la stânga, inclusiv cu λ-producții. Programul determină tabela de analiză sintactică asociată și decide dacă gramatica dată este $LL(1)$ (se va afișa mesajul DA sau NU, după caz). În caz afirmativ, programul permite citirea unui număr oarecare de șiruri peste alfabetul terminalilor. Pentru fiecare șir terminal se determină, pe baza tabelei de analiză sintactică obținută, dacă este în limbajul generat de gramatica respectivă iar în caz afirmativ se afișează derivarea sa stângă (o succesiune de numere, fiecare număr reprezentând numărul producției aplicate). (pot face echipă 2 studenți)

8. Să se scrie un program care implementează algoritmul pentru gramatici $SLR(1)$. Programul primește la intrare elementele unei gramatici independente de context oarecare, inclusiv cu λ-producții. Programul determină tabela de analiză sintactică asociată și decide dacă gramatica dată este $SLR(1)$ (se va afișa mesajul DA sau NU, după caz). În caz afirmativ, programul permite citirea unui

număr oarecare de șiruri peste alfabetul terminalilor. Pentru fiecare șir terminal se determină, pe baza tabelei de analiză sintactică obținute, dacă este în limbajul generat de gramatica respectivă iar în caz afirmativ se afișează derivarea sa stângă (o succesiune de numere, fiecare număr reprezentând numărul producției aplicate). (pot face echipă 2 studenți)

9. Să se scrie un program care implementează algoritmul pentru gramatici $LR(1)$. Programul primește la intrare elementele unei gramatici independente de context oarecare, inclusiv cu λ-producții. Programul determină tabela de analiză sintactică asociată și decide dacă gramatica data este $LR(1)$ (se va afișa mesajul DA sau NU, după caz). În caz afirmativ, programul permite citirea unui număr oarecare de șiruri peste alfabetul terminalilor. Pentru fiecare șir terminal se determină, pe baza tabelei de analiză sintactică obținute, dacă este în limbajul generat de gramatica respectivă iar în caz afirmativ se afișează derivarea sa stângă (o succesiune de numere, fiecare număr reprezentând numărul producției aplicate). (pot face echipă 2 studenți)

10. Să se scrie un program care primește la intrare o gramatică independentă de context G. Pentru fiecare producție $A \to x$ se va calcula mulțimea $First(x \cdot Follow(A))$. Programul verifică dacă pentru orice 2 producții $A \to x, A \to y, x \neq y$, $First(x \cdot Follow(A)) \cap First(y \cdot Follow(A)) = \emptyset$. 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.

11. Sa se studieze specificatia pentru generatorul de parser-e Bison. Sa se exemplifice pentru gramatica sintaxei limbajului C++ sau Python (se vor considera minim două tipuri de variabile, minim 2 instrucțiuni, dintre care o instrucțiune if și una de ciclare).