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

DEADLINE: 29 APRILIE

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 4 STUDENȚI DIN ACEEAȘI GRUPĂ POT ALEGE UNA DINTRE TEMELE 1-3, 8-11, DAR ACEASTA VA FI FĂCUTĂ INDIVIDUAL. TEMELE 4-7 POT FI ALESE DE MAXIM 2 ECHIPE DE 2 PERSOANE

FIECARE STUDENT VA COMPLETA ÎN FIȘIERUL DIN FILES Proiectul ales FPC 2024.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 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.
2. Program care simuleaza functionarea unui translator stiva nedeterminist cu λ-tranzitii. Programul citeste elementele unui translator stiva nedeterminist cu λ-tranzitii oarecare (starile, starea initiala, starile finale, alfabetul de intrare, alfabetul de iesire, alfabetul stivei, simbolul initial al stivei, tranzitiile). Programul permite citirea unui nr oarecare de siruri peste alfabetul de intrare al translatorului. Pentru fiecare astfel de sir se afiseaza toate iesirile (siruri peste alfabetul de iesire) corespunzatoare (Atentie! pot exista 0, 1 sau mai multe iesiri pt acelasi sir de intrare).
3. Sa se implementeze cu ajutorul programului flex (Fast Lexical Analyzer) sau unul din variantele sale: jlex sau jflex un analizor lexical pentru un limbaj de programare la alegere. Pentru fiecare token recunoscut, se vor returna: șirul din fișierul analizat care corespunde token-ului (lexema), tipul token-ului curent, linia din fisierul de intrare pe care se afla token-ul curent, un mesaj de eroare atunci cand este intalnita o eroare lexicala.
4. Sa se scrie un program pentru implementarea algoritmului de analiza sintactica Earley. Programul primeste la intrare elementele unei gramatici independente de

context oarecare, inclusiv cu λ-productii. Programul accepta un numar oarecare de siruri peste alfabetul terminalilor. Pentru fiecare sir se creeaza si se afiseaza tabelele Earley corespondente pe baza cărora se precizează dacă șirul respectiv este acceptat sau nu. În cazul în care șirul apartine limbajului generat de gramatică, afiseaza derivarile acelui șir plecand din simbolul de start. (pot face echipa 2 persoane, daca se obtin si derivarile).

5. Sa se scrie un program care implementeaza algoritmul pentru gramatici $LL(1)$. Programul primeste la intrare: elementele unei gramatici independente de context, nerecursiva la stanga, oarecare. Programul determina tabela de analiza sintactica asociata si decide daca gramatica data este $LL(1)$. In caz afirmativ, programul permite citirea unui nr oarecare de siruri peste alfabetul terminalilor. Pentru fiecare sir terminal se determina, pe baza tabelei de analiza sintactica obtinuta, daca este in limbajul generat de gramatica respectiva iar in caz afirmativ se afiseaza derivarea sa stanga (o succesiune de numere, fiecare numar reprezintand numarul productiei aplicate) (pot face echipa 2 persoane)
6. Sa se scrie un program care implementeaza algoritmul pentru gramatici $SLR(1)$. Programul primeste la intrare elementele unei gramatici independente de context oarecare. Programul determina tabela de analiza sintactica asociata si decide daca gramatica data este $SLR(1)$. In caz afirmativ, programul permite citirea unui nr oarecare de siruri peste alfabetul terminalilor. Pentru fiecare sir terminal se determina, pe baza tabelei de analiza sintactica obtinuta, daca este in limbajul generat de gramatica respectiva iar in caz afirmativ se afiseaza derivarea sa dreapta (o succesiune de numere, fiecare numar reprezintand numarul productiei aplicate). (pot face echipa 2 persoane)
7. Sa se scrie un program care implementeaza algoritmul pentru gramatici $LR(1)$. Programul primeste la intrare elementele unei gramatici independente de context oarecare. Programul determina tabela de analiza sintactica asociata si decide daca gramatica data este $LR(1)$. In caz afirmativ, programul permite citirea unui nr oarecare de siruri peste alfabetul terminalilor. Pentru fiecare sir terminal se determina, pe baza tabelei de analiza sintactica obtinuta, daca este in limbajul generat de gramatica respectiva iar in caz afirmativ se afiseaza derivarea sa dreapta (o succesiune de numere, fiecare numar reprezintand numarul productiei aplicate) (pot face echipa 2 persoane).
8. Sa se studieze specificatia pentru generatorul de parser-e Bison. Sa se exemplifice pentru gramatica sintaxei limbajului C++ (se vor considera minim două tipuri de variabile, minim 2 instrucțiuni, dintre care o instrucțiune if și una de ciclare).
9. Sa se studieze specificatia pentru generatorul de parser-e Bison. Sa se exemplifice pentru gramatica sintaxei limbajului Lua (se vor considera minim două tipuri de variabile, minim 2 instrucțiuni, dintre care o instrucțiune if și una de ciclare).
10. Sa se studieze specificatia pentru generatorul de parser-e Bison. Sa se exemplifice pentru gramatica sintaxei limbajului Python (se vor considera minim două tipuri de variabile, minim 2 instrucțiuni, dintre care o instrucțiune if și una de ciclare).

11. Sa se scrie un program care primește la intrare o gramatică independentă de context G. Pentru fiecare productie $A \to x$ se va calcula multimea $First(x \cdot Follow(A))$. 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 fisier text care va contine functiile (in C sau C++) corespunzătoare fiecărui neterminal din G, folosind algoritmul recursiv descendent pentru gramatica data.