1.Program care simuleaza functionarea unui automat finit nedeterminist cu lambda-tranzitii. Programul citeste (dintr-un fisier sau de la consola) elementele unui automat finit nedeterminist cu lambda-tranzitii oarecare (starile, starea initiala, starile finale, alfabetul automatului, tranzitiile). Programul permite citirea unui nr oarecare de siruri peste alfabetul de intrarea al automatului. Pentru fiecare astfel de sir se returnează DA sau NU, după cum șirul respectiv aparține sau nu limbajului acceptat de automat.

2.Program care simuleaza functionarea unui translator finit nedeterminist cu lambda-tranzitii. Programul citeste (dintr-un fisier sau de la consola) elementele unui translator finit nedeterminist cu lambda-tranzitii oarecare (starile, starea initiala, starile finale, alfabetul de intrare, alfabetul de iesire, tranzitiile). Programul permite citirea unui nr oarecare de siruri peste alfabetul de intrarea 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. Să se scrie un program care primește la intrare elementele unui automat finit nedeterminist cu lambda-tranzitii într-un fisier (starile, starea initiala, starile finale, alfabetul automatului, tranzitiile). Programul va implementa algoritmul din cursul 2 prin care se obține un automat finit determinist echivalent. Se vor afișa sub formă de graf elementele AFD obținut.
(pot face echipa 2 persoane)

4. Sa se scrie un program care primeste la intrare elementele unei expresii regulate (alfabetul expresiei, expresia propriu -zisa (in forma prefixata sau infixata - adica forma naturala), care contine 3 tipuri de operatori: reuniune, concatenare si iteratie Kleene (*)). Sa se determine un automat finit nedeterminist cu lambda-tranzitii, folosind algoritmul Thompson,care recunoaste acelasi limbaj ca cel descris de expresia regulata. Programul afiseaza si graful care corespunde noului automat.

5.Program care simuleaza functionarea unui automat pushdown nedeterminist cu lambda-tranzitii. Programul citeste (dintr-un fisier sau de la consola) elementele unui automat push-down nedeterminist cu lambda-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.

6.Program care simuleaza functionarea unui translator stiva nedeterminist cu lambda-tranzitii. Programul citeste (dintr-un fisier sau de la consola) elementele unui translator stiva nedeterminist cu lambda-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).

7. 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.

8.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, iar 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).

9. Sa se scrie un program care primeste la intrare $k \geq 1$ și elementele unei gramatici independente de context oarecare, G, inclusiv cu λ-productii, pentru care:
a) calculeaza multimile $First_k(X)$, $Follow_k(X)$, pentru fiecare simbol X terminal sau neterminal
b) elimina recursivitatea la stanga (pentru gramatici fără λ-producții)
c) factorizează stânga gramatica G.

10. 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)

11. 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)

12. 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).

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

14. Sa se studieze specificatia pentru generatorul de parser-e Bison. Sa se exemplifice pentru gramatica sintaxei unui limbaj/sublimbaj (se vor considera minim două tipuri de variabile, minim 2 instrucțiuni, dintre care o instrucțiune if și una de ciclare) pentru Java.

15. Sa se studieze specificatia pentru generatorul de parser-e Bison. Sa se exemplifice pentru gramatica sintaxei unui limbaj/sublimbaj (se vor considera minim două tipuri de variabile, minim 2 instrucțiuni, dintre care o instrucțiune if și una de ciclare) pentru Python.

16. Sa se studieze specificatia pentru generatorul de parser-e Bison. Sa se exemplifice pentru gramatica sintaxei unui limbaj/sublimbaj (se vor considera minim două tipuri de variabile, minim 2 instrucțiuni, dintre care o instrucțiune if și una de ciclare) pentru Prolog sau Haskell.

17. Sa se scrie un program care genereaza un fisier text care va contine functiile (in C sau C++) pentru algoritmul recursiv descendent pentru o gramatica data. La intrare programul primeste gramatica respectiva, care se presupune ca este de tip LL(1). Pentru fiecare productie A->x trebuie sa calculati multimea First(xFolow(A)).