Deadline SAPTAMANA 1.03 – 5.03.2021

1.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).

2. 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 determinist (sa se afiseze elementele sale) care recunoaste acelasi limbaj ca cel descris de expresia regulata, folosind algoritmul de la curs. Programul afiseaza si graful care corespunde noului automat (grafic 1 punct). (2 persoane)

3. 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 (sa se afiseze elementele sale) care recunoaste acelasi limbaj ca cel descris de expresia regulata. Programul afiseaza si graful care corespunde noului automat (grafic 1 punct).

4.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).

5. Sa se scrie un analizor lexical pentru limbajul C. Scrieti analizorul sub forma unei functii care returneaza: tipul token-ului curent, lungimea sirului corespunzator din fisierul de intrare, linia din fisierul de intrare pe care se afla token-ul curent, pointer catre primul caracter al token-ului curent, un mesaj de eroare atunci cand este intalnita o eroare lexicala. Functia este apelata din programul principal, in care este citit un fisier de intrare care va fi scanat cu ajutorul acestei functii, astfel incat sa se afiseze toti token-ii care apar in fisierul de intrare. Atunci cand este apelata, functia de scanare:
-incepand de la pointerul curent (care initial indica catre primul caracter al fisierului de intrare) sare peste un nr de caractere egal cu lungimea token-ului anterior (initial aceasta lungime este 0);
-sare peste spatii, tab-uri, linii noi, pana intalneste primul caracter diferit de acestea; seteaza pointerul curent astfel ca sa indice catre acest caracter;
-identifica token-ul curent, ce corespunde sirului ce incepe cu caracterul depistat la pasul anterior; determina tipul acestuia si lungimea sirului corespunzator;
-In cazul in care este intalnita o eroare lexicala, semneleaza aceasta printr-un mesaj, scaneaza fisierul de intrare in continuare, pana gaseste primul caracter de tip spatiu, linie noua, tab, seteaza pointerul curent catre acest caracter, seteza lungimea token-ului curent cu 0 (in felul acesta programul va afisa in continuare token-ii urmatori, fara sa se opreasca la prima eroare intalnita).
-se opreste cu scanarea cand a intalnit sfarsitul fisierului de intrare.

6. Sa se scrie un analizor lexical pentru limbajul C++. Cerințele sunt aceleași ca la punctul 5).

7. Sa se scrie un analizor lexical pentru limbajul Python. Cerințele sunt aceleași ca la punctul 5).

8. Sa se scrie un analizor lexical pentru limbajul Prolog. Cerințele sunt aceleași ca la punctul 5).

9. Sa se scrie un analizor lexical pentru Haskell. Cerințele sunt aceleași ca la punctul 5).

10. Sa se studieze programul flex (Fast Lexical Analyzer) sau unul din variantele sale: jlex sau jflex. Sa se ilustreze cu exemple pentru limbajul C (a se vedea și cerințele de la punctul 5).

11. Sa se studieze programul flex (Fast Lexical Analyzer) sau unul din variantele sale: jlex sau jflex. Sa se ilustreze cu exemple pentru limbajul C++ (a se vedea și cerințele de la punctul 5).

12. Sa se studieze programul flex (Fast Lexical Analyzer) sau unul din variantele sale: jlex sau jflex. Sa se ilustreze cu exemple pentru limbajul Python (a se vedea și cerințele de la punctul 5).

13. Sa se studieze programul flex (Fast Lexical Analyzer) sau unul din variantele sale: jlex sau jflex. Sa se ilustreze cu exemple pentru limbajul Prolog (a se vedea și cerințele de la punctul 5).

14. Sa se studieze programul flex (Fast Lexical Analyzer) sau unul din variantele sale: jlex sau jflex. Sa se ilustreze cu exemple pentru limbajul Haskell (a se vedea și cerințele de la punctul 5).