LABORATOR, PROIECTUL 1 (fiecare student va alege un singur proiect)

DEADLINE: 18 MARTIE 2024 (grupa 461), 19 martie (grupele 462, 463, 464)

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 GRUPĂ POT ALEGE UNA DINTRE TEMELE 1-8, DAR ACEASTA VA FI FĂCUTĂ INDIVIDUAL. TEMELE 9 ȘI 10 POT FI ALESE DE MAXIM O ECHIPĂ DE 2 PERSOANE

FIECARE GRUPĂ VA CREA UN FIȘIER ÎN FILES CU NUMELE 46x_tema_aleasa pe care îl va completa fiecare student din grupa respectivă:

NUME ȘI PRENUME nr-temei

Veți înărca proiectul în assignment-ul PROIECT 1 LABORATOR SERIA 46 sub forma unui fișier text pe care îl veți verifica cu Turnitin (opțiunea Turnitin este activă)

1) Să se scrie un analizor lexical pentru limbajul C. Scrieți analizorul sub forma unei funcții care returnează: tipul token-ului curent, lungimea șirului corespunzător din fișierul de intrare, linia din fișierul de intrare pe care se află token-ul curent, pointer către primul caracter al token-ului curent, un mesaj de eroare atunci când este întâlnită o eroare lexicală. Funcția este apelată din programul principal, în care este citit un fișier de intrare care va fi scanat cu ajutorul acestei funcții, astfel încât să se afișeze toți token-ii care apar în fișierul de intrare. Atunci când este apelată, funcția 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;

-Returnează numărul liniei din fișierul de intrare;

-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, seteaza 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.

Pentru fișierul de intrare următor:

```
1:
2:
3: void main(){    /* bbb*b
4:             bb  */
5:    int beta;
6:      beta+++ beta;
7:        A == b + 13.78.34
```

Se va returna:

‘void’, key_word; 4; linia 3

‘main’ key_word; 4; linia 3

‘(‘, delimitator; 1; linia 3

‘)’, delimitator; 1, linia 3

‘{’, delimitator; 1, linia 3

‘bbbb*bbb’, comment; 6; liniile 3-4

‘int’, key_word; 3, linia 5

‘beta’, identifier; 4; linia 5

‘;’, delimitator; 1, linia 5

‘beta’, identifier; 4; linia 6

‘++’, operator; 2, linia 6

‘+’, operator; 1, linia 6

‘beta’, identifier; 4; linia 6

‘;’, delimitator; 1, linia 6

‘A’, identifier; 1, linia 7

‘==’, operator; 2, linia 7

‘b’, identifier; 1, linia 7

‘+’, operator; 1, linia 7

’13.78’, float_constant; 5; linia 7

’.34’, float_constant; 3; linia 7

2. Aceeași temă ca la 1), dar pentru limbajul C++

3. Aceeași temă ca la 1), dar pentru limbajul Python

4. Aceeași temă ca la 1), dar pentru limbajul Java

5. Aceeași temă ca la 1), dar pentru un alt limbaj decât cele de la 1)-4)

6. 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 (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.

7. Program care simuleaza functionarea unui translator finit nedeterminist cu lambda-tranziții. Programul citește (dintr-un fișier sau de la consolă) elementele unui translator

finit nedeterminist cu lambda-tranziții 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 pentru acelasi șir de intrare).

8. 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.

9. 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 (*)).
a) Sa se determine forma postfixată a expresiei cu algoritmul Shunting Yard.
b) Folosind forma postfixată a expresiei, să se determine un automat finit nedeterminist cu lambda-tranzitii, folosind algoritmul Thompson, care recunoaste același limbaj ca cel descris de expresia regulată. Programul afisează și graful care corespunde noului automat.
(pot face echipă 2 persoane)

10. Sa se scrie un program care primeste la intrare elementele unei expresii regulate (alfabetul expresiei, expresia propriu -zisa (infixata - adica forma naturala), care contine 3 tipuri de operatori: reuniune, concatenare si iteratie Kleene (*)).
a) Sa 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 recunoaste acelasi limbaj ca cel descris de expresia regulata. Programul afisează și graful care corespunde noului automat.
(pot face echipă 2 persoane)