Zadanie č. 4

Vo zvolenom (podľa seba) programovacom jazyku zostavte program, ktorý:

1. Pre zadaný regulárny výraz zostrojí NKA / DKA, ktorý akceptuje jazyk popísaný daným výrazom.
2. Následne pre zadaný reťazec od používateľa vypíše informáciu o tom, či zadaný reťazec patrí / nepatrí do jazyka daného regulárnym výrazom.

Vstupy a výstupy

Regulárny výraz je zapísaný v súbore, ktorý predstavuje prvý argument programu pri jeho spúšťaní.

NKA / DKA ekvivalentný regulárnemu výrazu sa zapíše do súboru, ktorý predstavuje druhý argument programu pri jeho spúšťaní (to, či uložíte NKA alebo DKA nechám na Vás).

Ak by sa teda Váš program volal **zadanie4.exe**, program by sa spúšťal:

```
zadanie4.exe regex.txt KA.txt
```

kde regex.txt obsahuje na prvom riadku regulárny výraz, ktorý sa má akceptovať a KA.txt je meno súboru, kam sa zapíše výsledný zostrojený (NKA)DKA, ktorý akceptuje jazyk daný regulárnym výrazom.

Program po spustení vyzve používateľa, aby zadal nejaký reťazec nad vstupnou abecedou výsledného automatu a zistí, či jazyk popísaný regulárnym výrazom reťazec obsahuje alebo nie - informáciu o tom vypíše na obrazovku.

Implementácia algoritmov

Požaduje sa vaša samostatná práca, t.j. výsledný program musí byť výsledkom vašej samostatnej práce. Kopírovanie zdrojových kódov z internetu, prípadne od iných študentov bude hodnotené ako plagiátorstvo a v zmysle platného študijného poriadku hodnotené známkou FX. Taktiež použité algoritmy musia byť výsledkom vašej vlastnej implementácie, t.j. vytvorenie DKA pre daný regulárny výraz má byť výsledok vašej práce, vaša implementácia nejakého rozumného algoritmu.

Čo používať môžete je prístup, pri ktorom si pre gramatiku generujúcu regulárne výrazy zostrojíte jej **parser** - v takom prípade môžete použiť programy generujúce parsery, napr. Bison - a takýto parser použiť ako súčasť svojho vypracovaného zadania. Alebo si ten parser naprogramujete vy.

Deadline zadania

Zadanie odovzdajte do AIS-u do príslušného miesta odovzdania - odovzdajte zdrojový kód. Deadline je 12. máj 2019, 23:59:59.

FORMÁT VSTUPNÉHO SÚBORU S REGULÁRNYM VÝRAZOM

Na prvom riadku súboru sa nachádza regulárny výraz. Tento regulárny výraz popisuje jazyky nad abecedou, ktorú tvoria **malé a veľké písmená anglickej abecedy**. Obsahuje teda znaky predstavujúce **malé a veľké písmená** anglickej abecedy, ďalej obsahuje znak reprezentujúci **prázdny reťazec** - v našom prípade ním je **medzera**. Ďalej môže obsahovať **ľavú a pravú zátvorku, t.j. (,)** a symboly $*, |$ pre iteráciu a zjednotenie. Zreťazenie nemá špeciálny symbol, zreťazené sú tie skupiny symbolov, ktoré idú za sebou. Regulárny výraz je definovaný v zmysle prednášok, t.j.:

Nech T je abeceda malých a veľkých písmen anglickej abecedy. Potom:

- (medzera) je regulárny výraz popisujúci jazyk $\{\varepsilon\}$
- a, pre $a \in T$ je regulárny výraz popisujúci jazyk $\{a\}$
- Ak R_1 a R_2 sú regulárne výrazy popisujúce jazyky pomenované R_1 a R_2, potom $(R_1|R_2)$ je regulárny výraz popisujúci zjednotenie jazykov $R_1 \cup R_2$

- Ak R_1 a R_2 sú regulárne výrazy popisujúce jazyky pomenované R_1 a R_2, potom (R_1R_2) je regulárny výraz popisujúci zreťazenie jazykov R_1R_2
- Ak R je regulárny výraz popisujúci jazyk R, potom (R^*) je regulárny výraz popisujúci iteráciu R^* jazyka R.
- iné regulárne výrazy nie sú.

Operátory zjednotenia a zreťazenia môžeme považovať za asociatívne, preto kvôli väčšej prehľadnosti môžeme vynechať niektoré zátvorky, t.j. napríklad namiesto zjednotenia $(a|(b|(c|d)))$ napíšeme $(a|b|c|d)$, resp. namiesto zreťazenia $(a(b(cd)))$ napíšeme $(abcd)$.

FORMÁT VÝSTUPNÉHO SÚBORU S NKA (DKA)
Viď. zadanie č. 2, resp. zadanie č. 3.

GRAMATIKA GENERUJÚCA JAZYK POPISUJÚCI REGULÁRNE VÝRAZY

Pre jednoduchosť uvádzam príklad gramatiky, ktorá generuje regulárne výrazy, aké sa môžu vyskytnúť v zadaní - ak viete spraviť DKA k regulárnemu výrazu bez tejto informácie, tak túto stranu môžete ignorovať...:

Gramatika $G = (N, T, P, \text{Regex})$, ktorá obsahuje:

4 neterminály, $N = \{$ Regex, Term, Factor, Base $\}$.

5 terminálov $T = \{$ Char, $*$, $|$, $($, $)\}$.
(T.j. slovo Char, hviezdička, zvislá čiara, ľavá a pravá zátvorku.)

Pravidlá P:

1. Regex $\rightarrow$ Term
2. Regex $\rightarrow$ Term | Regex
3. Term $\rightarrow$ Factor Term
4. Term $\rightarrow$ Factor
5. Factor $\rightarrow$ Base
6. Factor $\rightarrow$ Base $*$
7. Base $\rightarrow$ Char
8. Base $\rightarrow$ (Regex)

Poznámka č. 1 V pravidle č. 2 je symbol | terminál, nie oddeľovač 2 pravidiel s rovnakou ľavou stranou!

Poznámka č. 2 Terminál Char je lexéma (viď. prednáška o lexikálnej analýze), predstavujúca elementárne regulárne výrazy (1 znak abecedy alebo prázdny reťazec), t.j. v našom prípade:

- Malé písmená anglickej abecedy
- Veľké písmená anglickej abecedy
- Znak popísujúci prázdny reťazec (v našom prípade ním bude medzera).

Príklad č. 1: Regulárny výraz je daný vo vstupnom súbore ako

```
((a(a*)b)|(a(b*)b))
```

Vytvorí sa teda NKA pre regulárny výraz $((a(a^*)b)|(a(b^*)b))$. Následne sa vytvorí DKA k nemu ekvivalentný a program vyzve používateľa, aby zadal nejaký reťazec: Napríklad pre reťazec *aaaab* by program vypísal, že príslušný DKA takéto slovo **AKCEPTUJE.** Pre reťazec *aabab* by vypísal, že slovo sa **NEAKCEPTUJE**.

Príklad č. 2: Regulárny výraz je daný vo vstupnom súbore ako

```
(((ba)*)( |c)(b*))
```

Zostrojí sa teda NKA pre regulárny výraz, ktorý by sa dal cez klasické označenie zapísať ako $(((ba)^*)(\varepsilon|c)(b^*))$. Následne sa vytvorí DKA k nemu ekvivalentný a program vyzve používateľa, aby zadal nejaký reťazec: Napríklad pre reťazec *babacbbb* by program vypísal, že príslušný DKA takéto slovo **AKCEPTUJE.** Pre reťazec *babaccbb* by program vypísal, že príslušný DKA takéto slovo **NEAKCEPTUJE.** POZOR! Regulárny výraz obsahuje **medzeru** (ktorá v tomto textovom zápise reprezentuje prázdny reťazec).

Príklad č. 3: Regulárny výraz je v prvom riadku súboru ako

```
(((ab)|(ba)|(bb)|(aa))((a|b)*))
```

T.j. regulárny výraz popisuje všetky reťazce zo znakov a, b dĺžky aspoň 2. Napríklad pre reťazec ab by program vypísal, že príslušný DKA takéto slovo **AKCEPTUJE.** Pre reťazec a by vypísal, že ho **NEAKCEPTUJE.**

Príklad č. 4: Regulárny výraz je (pozor, obsahuje medzeru!)

```
((A|B|C)(((a|b|c)*)| )(A|B|C))
```

Výraz teda popisuje reťazce z písmen A, B, C, a, b, c také, ktoré začínajú a končia veľkým písmenom a medzi nimi sa môže nachádzať ľubovoľná postupnosť z malých písmen. Reťazec $AbacB$ by mal príslušný DKA **AKCEPTOVAŤ.** Reťazec $ABacB$ by takýto DKA **NEAKCEPTOVAL.**

Príklad č. 5: Regulárny výraz vo vstupnom súbore:

```
(((I|i)(f|(nt)))|((E|e)((nd)|(lse)))|((B|b)(egin)))
```

Výraz teda popisuje reťazce If, if, Int, int, End, end, Else, else, Begin, begin. Následne sa vytvorí DKA k nemu ekvivalentný a program vyzve používateľa, aby zadal nejaký reťazec: Napríklad pre reťazec *End* by program vypísal, že príslušný DKA takéto slovo **AKCEPTUJE.** Reťazec *enD* by takýto DKA **NEAKCEPTOVAL.**