Problema Clasificator

Fişier de intrare	`input.txt`
Fişier de ieşire	`output.txt`

Poveste

Cerință

Se dă alfabetul format din literele mici ale alfabetului englez (a-z) și două limbaje finite, `Accept` și `Fail`, cu intersecția vidă. Să se construiască o expresie regulată sau un automat finit determinist care să accepte cuvintele din limbajul `Accept` și să le respingă pe cele din limbajul `Fail`. Dacă un cuvânt nu aparține unuia din limbaje, nu contează cum este tratat de expresia regulată sau de automat.

Date Intrare

Datele vor fi citite din fișierul **input.txt**.

Pe prima linie se găsesc trei numere, `cnt_accept` (cardinalul limbajului `Accept`), `cnt_fail` (cardinalul limbajului `Fail`) și `len_string` (lungimea fiecărui cuvânt, **toate cuvintele au aceeași lungime**). Pe următoarele `cnt_accept` linii se găsesc cuvintele din limbajul `Accept`. Pe ultimele `cnt_fail` linii se găsesc cuvintele din limbajul `Fail`.

Date Ieșire

Datele vor fi scrise în fișierul **output.txt**.

Puteți alege între a construi o expresie regulată (regex) sau un automat finit determinist (DFA):

Regex

Dacă răspunsul pentru un test este formulat ca o expresie regulată, prima linie trebuie să conțină cuvântul "`regex`", iar a doua linie trebuie să conțină expresia regulată. Ea poate fi formată din următoarele caractere:

- () – paranteze, pentru gruparea termenilor
- * – Kleene star
- + – Kleene star, cu minim o apariție
- | – reuniune
- a-z – litere
- . – wildcard

DFA

Dacă răspunsul pentru un test este formulat ca un automat finit determinist, prima linie trebuie să conțină cuvântul "`dfa`".

A două linie trebuie să conțină trei numere: numărul de stări n ale automatului, numărul de stări finale f ale automatului și indicele stării inițiale (indexat de la 0).

A treia linie trebuie să conțină f numere, indicii stărilor finale.

Următoarele n linii descriu matricea de tranziție a automatului (notată cu M), fiecare conținând $|\Sigma| = 26$ numere. Al j-lea număr ($0 \leq j < |\Sigma|$) de pe linia i descrie indicele stării în care ajunge automatul dacă se află în starea i cu simbolul $\Sigma[j]$ la intrare. ($\delta(i, \Sigma[j]) = M[i, j], \Sigma[0] = a, \Sigma[25] = z$)

Restricții

- `len_string` ≤ 100

#	Punctaj	Restricţii
1	76	`cnt_accept` + `cnt_fail` ≤ 100
2	24	`cnt_accept` + `cnt_fail` ≤ 400

Formula pentru calculul punctajului pe un test este:

$$2 \cdot \frac{\texttt{ans\_ref}}{\texttt{ans\_ref} + \texttt{ans\_participant}}$$

Unde `ans_ref` este o soluție a echipei. Dacă respunsul este dat sub forma unei expresii regulate, `ans_*` este lungimea expresiei. Altfel, este numărul de stări din automat (n). Scorul maxim este de **150** de puncte, din care **50** sunt bonus.

Exemple

input.txt	output.txt
3 2 4 baed bece bace bacd aace	regex b.(ed|.e)
12 7 9 ccaccccac acaccccac caaccccac abaccccac ccacccbc ccacbccbc ccacbbcbc ccacabcbc ccbccccac ccbccccbc ccacbccac ccbacccac ccbcbccbc ccabcccac ccbbbccbc ccacccbb ccbccccab abacbccac ccbaababc	regex .(...a|cacb|..(a|c)c).c.c
3 6 8 sureties baronies rarefies barflies sardines carbides safeties rareties bakeries	regex .(a..(f|n)|u...)...

Explicații

Puteți găsi mai jos câteva expresii regulate corecte pentru primul exemplu:

- b.(ed|ce)
- baed|bece|bace
- bace|.(ae|ec).
- bae.|bace|be..
- ..ed|(be|ba)ce
- ba(ed|ce)|be..
- bace|.(aed|ece)
- b(aed|ece|ace)
- bace|b(ae|ec).
- b.*e.*

Pentru mai multe exemple, inclusiv cu DFA, puteți consulta directoarele outputs_regex și outputs_dfa din checker.

Precizări

- Checker-ul este disponibil pe gitlab
- Pentru expresii regulate, puteți folosi regex101 pentru a găsi rapid greșeli în răspuns.
- Tema va fi testată pe VMChecker. Limita de timp per test este de 60 de secunde, iar pentru toată tema limita este de 400 de secunde. Nu există o limită de memorie pentru heap. Mașina virtuală are 4 GB de memorie alocați.
- `Makefile`-ul trebuie să se afle neapărat în rădăcina arhivei incărcate pe Moodle.
- Dimensiunea stivei este mărită de checker (doar în timpul rulării lui) la 256 MB. Dacă vă testați independent de checker programul, puteți să vă măriți dimensiunea stivei în sesiunea curentă de terminal cu `ulimit -s 262144`. Puteți să verificați dimensiunea stivei cu `ulimit -a | grep stack`.
- Checker-ul poate fi invocat astfel: `./checker checker_info.txt ../src/`, unde `checker_info.txt` este un fișier care descrie toate testele pe care va fi rulată sursa. Checker-ul va încerca să apeleze `make` din folder-ul dat ca argument (`../src/`), deci trebuie neapărat să existe un `Makefile` acolo cu regulile `build`, `run` și `clean`.
- Tentativele de a păcăli checker-ul vor fi pedepsite.
- Dacă vreți să implementați rezolvarea într-un limbaj care nu este suportat de imaginea de docker a checker-ului, postați un mesaj pe forum.