Problem 1

JSON parser rađen na 4. auditornim vježbama posjeduje jedan propust, naime moguće je ostaviti zarez na zadnjem mjestu kod liste parova. Takav ulaz će parser označiti korektnim, iako ne bi trebao biti.

Primjer:

{ a : 12, b : 12,}

Za ovaj problem bi trebali:

1. Napisati kratko objašnjenje zašto data gramatika smatra ovakav ulaz tačnim,
2. prepraviti gramatiku tako da ulaz sa zadnjim tokenom bude nevalidan,
3. korigovati kod tako da oslikava novu gramatiku.

Spomenuti primjer koristi recursive descent parser.

Problem 2

Data je gramatika kalkulatora:

$$E \to TE'$$
$$E' \to +TE' | -TE' | \epsilon$$
$$T \to FT'$$
$$T' \to *FT' | \epsilon$$
$$F \to -E | (E) | num | id$$

Za datu gramatiku potrebno je napisati prediktivni parser, primjenom metoda koji je rađen na predavanju 9.

Pri izradi koda studenti mogu tokenizaciju uraditi korištenjem flex alata. Priložiti dokument koji objašnjava postupak generisanja tabele za prediktivni parser.

Parser je potrebno implementirati ručno. Odabir struktura podataka za ovaj problem se ostavlja studentu. Program treba na kraju ulaza da ispiše poruku *Ulaz je validan* ili *Ulaz nije validan* u zavisnosti od toga da li je ulaz zadovoljio gramatiku i da li je parser konzumirao sve tokene sa ulaza.

Predaja zadaće

Za svaki problem potrebno je napraviti zaseban direktorij sa imenima problem1, problem2, i td. Unutar direktorija se nalaze sve potrebne datoteke za kompajliranje programa. Svaki problem mora imati svoj Makefile koji će komandom make kompajlirati program i napraviti jedan izvršni file koji će nositi ime kao

i problem, dakle problem1, problem2 i td. Na primjer, naredna sekvenca shell komandi treba biti validna:

```
cd problem1
make
./problem1 < input_fajl
Ulaz je validan.
```

Program će koristit standardni ulaz i izlaz. Primjer Makefilea koji prepoznaje flex možete pronaći u rješenjima laba 3 za primjere json i scanner.