AKT eksami alusosa (7. mai 2018)

Põhiosa eesmärgiks on koostada etteantud reeglitele vastav (loogika) avaldise grammatika kasutades ANTLRi vahendeid, moodustada abstraktne süntaksipuu (kasutades etteantud AST klasse) ning implementeerida AST väärtustamise (*eval*) ja väljatrüki (*prettyString*) meetodid.

Soovitatav ülesannete lahendamise järjekord on: 1. *eval*, 2. grammatika kirjeldus, 3. ASTi koostamine, 4. *prettyString*. *Eval* ei sõltu muudest osadest, aga ülejäänud kolm peab tegema antud järjekorras, samas ei pea grammatika olema ideaalne, et hakata ASTi koostama.

AST moodustatakse *LoogikaNode* alamklassidest. Iga grammatikas kirjeldatud avaldise tüübi jaoks on oma AST klass. Loogilise tehte jaoks on kolm eraldi klassi: *JaNode*, *VoiNode*, *VordusNode* (eraldi klassi NING jaoks pole, kuna AST määrab üheselt ära tehete järjekorra).

1. Avaldise väärtustame (LoogikaUtil.eval) (4p)

LoogikaUtil.java failis olev *eval* meetod peab väärtustama (*true/false*) etteantud ASTi, kasutades lisaks etteantud tõeväärtusmuutujate *map.*. Tingimusavaldise korral, kui avaldises ei esine MUIDU osa ning KUI avaldis on väär, tuleks tagastada *true*. Enne implementeerimist tutvu ka grammatika reeglitega, AST klassid vastavad grammatikas kirjeldatud avaldise tüüpidele.

2. Grammatika koostamine (Loogika.g4) (4p)

Avaldis võib olla muutuja, literaal, tingimusavaldis, loogiline tehe või sulgudega ümbritsetud avaldis.

- Muutuja koosneb ühest või rohkemast ladina suur- või väiketähest.
- Literaaliks võib olla **1** või **0**, 1 esindab tõest väärtust ja 0 väära väärtust.
- Tingimusavaldis algab võtmesõnaga **KUI**, millele järgneb avaldis, seejärel võtmesõna **SIIS**, millele omakorda järgneb avaldis. Seejärel võib tulla võtmesõna **MUIDU** ning avaldis. Tingimusavaldis on madalama prioriteediga kui loogiline tehe.
- Loogiline tehe koosneb kahest avaldisest, millede vahel on operaator (**NING**, **VOI**, **JA**, **=**), operaatorid on vasakassotsiatiivsed v.a. =, mis ei ole üldse assotsiatiivne (s.t. `a = b = c` ei ole lubatud). Kõige madalama prioriteediga on NING, seejärel kasvavas järjekorras VOI, JA, =. Sulud on kõige kõrgema prioriteediga. Pane tähele, et NING on madalama prioriteediga kui VOI, aga loogiline tähendus on tal sama kui JA-l (eval(*true* NING *true*) = *true*).
- Avaldiste vahel ja ümber võib olla suvaline arv tühikuid ja tabulaatoreid.

NB! Grammatika koostamisel mõtle, missugune grammatika aitab hiljem mõistlikult ASTi koostada.

Näited legaalsetest avaldistest:

```
1;  a;  kala JA 0;  1 = a VOI z VOI a = (1 JA a NING d);
KUI a = b SIIS 1 MUIDU b JA c;  KUI a SIIS KUI c SIIS e MUIDU 1
```

3. AST koostamine (LoogikaAst.parseTreeToAst) (5p)

Siin ülesandes tuleb teisendada ANTLRi abil moodustatud *parse*-puu abstraktseks süntaksipuuks. AST moodustamisel tuleb kasutada *LoogikaNode.java* staatilisi meetodeid (*lit* – literaali tipp, *var* – muutuja, *ja*, *voi*, *vordus*, *kuiSiis*).

NB! Ära unusta enne ASTi ülesande juurde asumist genereerida lekser ja parser (*generateGrammarSource*). ASTi klasse ise muuta ei tohi.

4. AST väljatrükk (LoogikaUtil.prettyString) (2p)

NB! Meetodit pole vaja realiseerida *KuiSiisMuidu* klassi jaoks!

See ülesanne eeldab, et nii grammatika koostamine ja AST moodustamine on tehtud. Ülesande eesmärgiks on moodustada ASTi pealt avaldisele vastav sõne nii, et kui seda sõne oma grammatikaga parsida, siis moodustuks esialgsega **samaväärne** AST (s.t. et AST kuju oleks sama, mitte ainult tõeväärtus oleks sama). Lisaks soovime minimiseerida sulgude arvu, kasutades ära prioriteete ja assotsiatiivsust.

Pea meeles, et tõene literaal tuleks kuvada kui `1` ja väär literaal kui `0`.

Ühe punkti saab ilma NING operaatorit kasutamata, arvestades ainult `JA, VOI, =` prioriteete ja assotsiatiivsust.

```
(a JA b) JA c -> a JA b JA c;   a VOI (b JA c) -> a VOI b JA c;    (a = b) JA d -> a = b JA d
A VOI (b VOI c) -> A VOI (b VOI c)
```

Kahe punkti jaoks peab lisaks sobivalt paigutama `JA/NING` operaatoreid. Võid proovida seda realiseerida optimaalselt, aga kuna meie avaldised ei ole väga pikad, saab seda teha ka jõumeetodiga – kõiki `JA/NING` kombinatsioone läbi proovides.

```
A JA (b VOI c) -> A NING b VOI c;  a JA (b JA c) -> a NING b JA c
```

Esitama peab failid: **Loogika.g4, LoogikaAst.java, LoogikaUtil.java.** Lisaks saab esitada kuni kolm faili.