TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Kirsti Tagam

E-kursuse „Programmeerimise alused II” rekursiooni temaatika küsimuste ja ülesannete loomine

Bakalaureusetöö (9 EAP)

Juhendaja: Eno Tõnisson, MSc

Tartu 2017

E-kursuse „Programmeerimise alused II“ rekursiooni temaatika küsimuste ja ülesannete loomine

Lühikokkuvõte:

Käesoleva bakalaureusetöö raames koostati rekursiooni temaatika ülesandeid ja küsimusi e-kursusele „Programmeerimise alused II”. Ülesannete sobivust analüüsiti magistriõppekava “Infotehnoloogia mitteinformaatikutele” üliõpilaste lahenduste, tulemuste ja tagasiside põhjal.

Võtmesõnad: „Programmeerimise alused II”, e-kursus, MOOC, rekursioon, , õppematerjal

CERCS: P175, Informaatika, süsteemiteooria

Creating Recursion Themed Study Materials for E-course „Introduction to Programming II”

Abstract:

The purpose of this thesis is to create recursion themed study materials for e-course „Introduction to programming II”. Materials were analysed according to the results, solutions and feedback of the master’s students from curriculum „Infotehnoloogia mitteinformaatikutele”.

Keywords: „Introduction to programming II”, e-course, MOOC, recursion, study material e-course

CERCS: P175, Informatics, systems theory

Sisukord

1. Sissejuhatus

E-kursused on maailmas üha levivam trend. Interneti vahendusel õppimine on tänapäeval populaarne ning teadmisi saab ammutada paljudest eri valdkondadest [1]. Kursuseid pakuvad ülikoolid, ettevõtted, mittetulundusühingud, institutsioonid jpt [2]. Vaba juurdepääsuga e-kursusi nimetatakse rahvusvaheliselt ka MOOCideks ehk *Massive Open Online Course* [3]. Tartu Ülikooli arvutiteaduse instituut pakub vaba juurdepääsuga e-kursusi 2014. aastast, mil loeti esimest korda kursust „Programmeerimisest maalähedaselt". Kuna osalejate ja lõpetanute arv on olnud stabiilselt kõrge kursusel „Programmeerimisest maalähedaselt" kui ka selle jätkukursusel „Programmeerimise alused", loodi ka viimasele jätkukursus „Programmeerimise alused II", millele käesolev töö keskendub. Kõikidel eelpool nimetatud kursustel käsitletakse programmeerimisõpet programmeerimiskeeles Python.

Erinevalt tavakursustest, kus õppurid käivad kohapeal õppimas ning saavad vahetut tagasisidet, toimuvad e-kursused osaliselt või täielikult veebipõhiselt. Iseseisev õppimine nõuab üldiselt rohkem aega ning visadust, et materjal omandada. Et üks e-kursus oleks edukas, peaksid olema õppematerjal, harjutused ning ülesanded arusaadavad ning asjakohased erinevate taustadega õppuritele. Rekursiooni temaatikat pole varasemalt Tartu Ülikooli MOOCides kohustusliku osana käsitletud, küll aga oli ta silmaringiteemana huvilistele kursusel "Programmeerimise alused". Programmeerimise õppimisel algajana on rekursioon üks keeruline teema, mis vajab korralikult süvenemist ning keskendumist.

Antud töö eesmärk on luua küsimusi ning ülesandeid e-kursusele „Programmeerimise alused II", mis aitaksid õppuritel rekursioonist paremini aru saada. See on üks kolmest kursuse põhiteemast kahemõõtmelise järjendi ja andmestruktuuride kõrval. Töö käigus luuakse 1 nelja küsimusega Moodle'i test ja 1 viie küsimusega Moodle'i test, 1 enesekontrolliküsimus, 5 programmeerimisülesannet ning 10 ülesannet eksami tarbeks. Materjale analüüsitakse õppurite tulemuste põhjal ning antakse hinnanguid ka soorituste ja tagasiside põhjal. Kuna töö autor on informaatika üliõpilane, on ta ise rekursiooni hiljuti õppinud ning oskab õppimise protsessi algaja vaatenurgast hinnata ning seda materjalide loomisel rakendada. Koostatud materjalid on loodud koostöös kursuse korraldajatega. Käesoleva töö autor osales rekursiooni temaatika Moodle'i testide, programmeerimisülesannete, enesekontrolli küsimuste, näidiseksami ning eksamiülesannete loomisel. Bakalaureusetöö autor koostas ja

seadistas Moodle'i keskkonnas rekursiooni temaatika teste ning Courses.cs.ut.ee keskkonnas enesekontrolli küsimuse.

Bakalaureusetöö koosneb kolmest osast. Esimeses osas antakse ülevaade MOOCidest maailmas üldisemalt ning Tartu Ülikooli arvutiteaduse instituudi poolt korraldavatest programmeerimisega seotud MOOCidest. Teises osas käsitletakse esmalt rekursiooni mõistet, selle olemust ning kasutusvaldkondi. Seejärel antakse ülevaade rekursiooni võimalikest õppimisvõimalustest ning õpetamistehnikatest. Kolmas osa keskendub e-kursuse „Programmeerimise alused II" rekursiooni temaatika materjalide loomisele ning nende analüüsile. Lisades on osa antud töö raames koostatud materjalidest, mida töös detailsemalt ei käsitleta, et bakalaureusetöö põhiosa ei muutuks liiga pikaks.

2. MOOCid

Käesolevas peatükis antakse ülevaade e-kursuste ehk MOOCide taustast, populaarsetest MOOCe pakkuvatest keskkondadest, MOOCidest maailmas ning Eestis. Täpsemalt käsitletakse Tartu Ülikooli Arvutiteaduse instituudi poolt pakutavaid MOOCe.

2.1 MOOCidest üldiselt

Esimene MOOC loodi 2008. aastal, kuid lai levik sai alguse 2012. aastal, kui Stanfordi ülikooli professorid Sebastian Thrun and Peter Norvig lõid kursuse „Introduction to Artificial Intelligence". Kursusele registreerus 190 000 inimest ning see oli selleks ajaks populaarseim infotehnoloogia teemaline MOOC [4]. 2016. aastaks kõige suurema osalejate arvuga kursus oli Stanfordi ülikooli poolt pakutav MOOC „Machine Learning", millel on aastast 2011 olnud kokku üle miljoni osaleja [5].

Tänaseks on maailmas erinevaid vaba juurdepääsuga e-kursuste keskkondi, näiteks edX, Coursera, mis on enamasti ingliskeelsed. Harvardi Ülikooli ja Massachusettsi Tehnoloogiainstituudi poolt läbi viidud uuringust selgub, et aastatel 2012-2016 on vastavate ülikoolide MOOCide populaarsus edX keskkonnas olnud tõusujoones. Nelja aasta jooksul on 2,4 miljonit inimest osalenud ühel või enamal kursusel [3].

MOOCe luuakse eesmärgil, et erinevate taustadega inimesed saaksid end arendada ilma kodust lahkumata. Enamasti on MOOCid tasuta, kuid mõned on tasulised, kui inimene soovib identifitseeritud sertifikaati tõestuseks sellele, et on kursuse läbinud. Ülikoolis õppides saab tudeng küsida õppejõududelt vahetut tagasisidet ning individuaalset abi. Kõikide MOOCide puhul ei ole võimalik õppuritele individuaalset tagasisidet anda, näiteks suure registeerunute arvuga MOOCide puhul. Selliste MOOCide õppurid peavad leidma vastused oma küsimustele suuremas osas iseseisvalt. Seega on veelgi olulisem, et kursuste materjalid ning ülesanded oleksid üheselt mõistetavad ning teemakohased. MOOCid on enamasti loodud ülikoolide poolt ning nendega tegelevad peamiselt õppejõud ning oma ala eksperdid. Üldiselt on e-kursuste lõpetajate arvud üsna väikesed võrreldes registreerunutega. Harvardi Ülikooli ja Massachusettsi Tehnoloogiainstituudi poolt tehtud uuringus selgus, kursusel, millele saab ligipääsu pärast registreerimist 7900 inimest, lõpetab keskmiselt 500 inimest sertifikaadiga ning 1500 inimest läbib pool või rohkem kursuse sisust [3].

Eestis on eestikeelsetest e-kursustest üks populaarsematest "Programmeerimisest maalähedaselt". Seda kursust pakub Tartu Ülikooli arvutiteaduse instituut alates 2014. aastast

ning lõpetanuid on tänaseks 4119 [6]. Kursusel käsitletakse programmeerimise õppimist programmeerimiskeeles Python.

2.2 Programmeerimise MOOCid Tartu Ülikoolis

Tartu Ülikooli arvutiteaduse instituut pakub programmeerimise MOOCe 2014. aasta talvest. Kõik pakutavad MOOCid on tasuta ning kursusel osalemiseks on vajalik eelnev registreerimine [6]. Käsitletavate MOOCide õppetöö toimub keskkondades Moodle ja Courses.cs.ut.ee, programmeerimisülesandeid lahendatakse vabavaralise programmi Thonny abil [7].

Esimene korraldatud MOOC oli "Programmeerimisest maalähedaselt", mida korraldatakse siiani. Kursus toimub kaks korda aastas, kevadel ja sügisel. Kursuse maht on 1 EAP ehk keskmiselt kulub inimesel sellele kokku ca 26 tundi, kursuse kestus on 4 nädalat. MOOCile „Programmeerimisest maalähedaselt" on oodatud registreerima kõik, kellel on huvi programmeerimise vastu, eelnev kogemus ei ole oluline. 11. mai 2017 seisuga on kursusele registreerunuid kokku 6444 ning läbinuid 4119 [6].

„Programmeerimisest maalähedaselt" jätkukursus on „Programmeerimise alused", mille registreerunutel on soovituslik omada eelnevat kogemust, kuid see ei ole registreerimiseks kohustuslik. MOOC „Programmeerimise alused" on siiani toimunud kevaditi alates 2016. aastast. Kursuse maht on 3 EAP ehk aine sooritamiseks kulub ligikaudu 78 tundi, kursuse kestus on 8 nädalat. 11. mai 2017 seisuga oli olnud kursusele registreerunuid kokku 3893 ning läbinuid 2103 [6].

„Programmeerimise alused" jätkukursus on „Programmeerimise alused II", millest on erinevaid variante: statsionaarne, segu statsionaarsest kursusest ja MOOCist ning MOOC. Käesolevas töös on analüüsitud kursuse teist varianti. Kursust loeti segu variandina esimest korda 2016. aasta sügisel testrühmaga. Kursus toimus ajavahemikus 20. oktoober – 15. detsember 2016. Testrühmaks olid magistriõppe „Infotehnoloogia mitteinformaatikutele" õppekava üliõpilased. „Infotehnoloogia mitteinformaatikutele" õppekavas on 6 EAP-line kursus „Programmeerimine", mis on põhiliselt „Programmeerimise alused" ja „Programmeerimise alused II" koos, kuid 17 üliõpilast kursuselt olid juba läbinud MOOCi „Programmeerimise alused" 3 EAP mahus. Neile pakuti kursuse „Programmeerimine" asemel 3 EAP-list kursust „Programmeerimise alused II", kus toimusid igal teisel nädalal auditoorsed loengud ja lisapraktikumid. Kursus oli jagatud perioodideks, igal perioodil oli õppuritele ette nähtud teatud ülesanded, töö toimus e-kursuse materjalidega. Esimesel

perioodil käsitleti põhiliselt kordamist, teisel mitmemõõtmelisi järjendeid, kolmandal rekursiooni ning neljandal andmestruktuure. Iga perioodi eest oli võimalik saada maksimaalselt 16 punkti. Moodle'i teste hinnati küsimuste arvu järgi, kuid kursuse lõpparvestuseks teisendati tulemused ümber 3-punkti skaalale. Programmeerimisülesannete puhul hilisemaid teisendusi ei tehtud. Iga perioodi lõpus oli Moodle's võimalik esitada muljeid ja ettepanekuid, mõne perioodi lõpus ka programmeerimisülesannete logifaile. Antud kursus andis korraldajatele võimaluse saada tagasisidet kursuse ülesehitusele, materjalidele ning üldisele korraldusele ning seda analüüsida enne kursuse MOOCi versiooni alustamist.

„Programmeerimise alused II" alustas MOOCina 3. aprillil 2017, kuhu on registreerunuid kokku 995. Kursusele „Programmeerimise alused II" registreerijad peaksid omama eelnevat kogemust kursuse „Programmeerimise alused" mahus, kuid see ei ole kursusele registreerimiseks kohustuslik. Kursuse maht on 3 EAP nagu ka kursusel „Programmeerimise alused" ning kursus kestab 8 nädalat.

3. Rekursioon

Käesolevas peatükis antakse ülevaade rekursioonist. Esmalt käsitletakse rekursiooni mõistet, selgitatakse selle olemust ja kasutusvaldkondi. Seejärel antakse ülevaade selle õppimisvõimalustest ning õpetamismeetoodikatest.

3.1 Rekursiooni mõiste

Arvutiteaduses nimetatakse rekursiooniks tavaliselt situatsiooni, kus funktsiooni definitsioonis on kasutatud parasjagu defineeritavat funktsiooni [8]. Rekursiooni kasutamise eesmärgiks on funktsiooni ennast välja kutsumise teel lahendada funktsioonile antud ülesande kergem variant ehk üks ülesanne jagatakse sarnasteks väiksemateks alamülesanneteks. Alamülesanded lahendatakse rekursiivselt ning kombineeritakse kokku põhiülesande lahenduseks. Sellist lähenemist probleemi lahendamisele nimetatakse ka „divide and conquer" meetodiks ehk „jaota ja valitse" [9]. Seda meetodit kasutades saab keerukaid ülesandeid lahendada vähema vaevaga, näiteks andmete sortimist ühildusmeetodil [9].

Rekursioonis peab vähemalt üks haru olema rekursiivse väljakutseta, et see funktsiooni töötamine mingil hetkel lõppeks, seda nimetatakse rekursiooni baasiks. Seda haru, mis kutsub funktsiooni rekursiivselt välja, nimetatakse rekursiooni sammuks [8].

Programmeerimisülesandeid saab rekursiivse lähenemise asemel lahendada ka tsüklite abil. Rekursiooni kasutamine tuleb tihti kasuks keerulisemate ülesannete lahendamise puhul, kus tsüklite kasutamine oleks keerulisem ning ebaefektiivsem. Rekursiooni kasutamine teeb tihti programmikoodi lühemaks ning selgemaks kui tsüklite kasutamine. Kõiki ülesandeid, mida saab lahendada tsüklitega, saab ka lahendada rekursiooniga [9]. Rekursiooni kasutatakse tihti graafide väljendamisel: erinevate hierarhiate, võrgustike jne korral.

Reaalelus võib rekursiooni kohata näiteks Vene rahvuslikuks sümboliks peetava Matrjoška nukkude näol. Ühe suure nuku seest leiab mitmeid väiksemaid samalaadseid nukke, milledest sisemine on kõige väiksem. Matrjoška puhul lõppeb rekursioon kõige väiksema nukuga, sest tema sees ei ole enam uut nukku, mida avada. Looduses võib rekursiooni näha puudena, üks puu koosneb paljudest väikestest puudest ehk hargnevatest okstest. Rekursiivsete programmidega saab moodustada fraktaleid. Arvutitega loodud fraktalid on omaette kunsti valdkond juba 1980-ndatest aastatest [10]. Fraktaleid luuakse kunstis arvutitarkvara abil, kuhu sisestatakse erinevaid võrrandeid ning valemeid. Nüüdseks on loodud internetti erinevaid galeerisid, kust saab tehtud töid vaadata [10, 11].

3.2 Rekursiooni õpe

Programmeerimist õpetatakse erinevates maailma haridusasutustes, ülikoolides õpetatakse näiteks arvutiteaduse, informaatika jne õppekavadel, üldiselt sisaldavad õppekavad ka rekursiooni käsitlemist. Programmeerimist, sealhulgas rekursiooni teemat, saab iseseisvalt õppida erinevatest allikatest, materjalidest, internetist. Interneti vahendusel on võimalik rekursiooni õppida ka eesti keeles. Internetti on üles laetud erinevate Eesti ülikoolide materjalid, on loodud MOOCe, eraisikute endi loodud portaale, veebilehekülgi. Rekursiooni saab õppida ka programmeerimisõpikutest, kus rekursiooni käsitletakse tavaliselt eraldi peatükis, näiteks on nii Jüri Kiho raamatus „Java programmeerimise aabits" [12]. Eesti keeles on enamus programmeerimisraamatuid rekursiooni õppimiseks programmeerimiskeelte Java ja Pythoni põhjal. Inglise keeles on võimalusi rekursiooni õppimiseks rohkem. Erinevaid õppematerjale ning abi õppimiseks leiab ülikoolide materjalide hulgast, foorumitest, programmeerimise õppimise kodulehekülgedelt, MOOCide saitidelt, videoportaalist YouTube jne.

Erinevates programmeerimise algõppe materjalides käsitletakse rekursiooni teemat enamasti lõpupoole, kui on eelnevalt selgeks õpitud funktsioonid, tsüklid jne. Näitena võib tuua õpikud „Python programming: an introduction to computer science" [9] ja „Java programmeerimise aabits" [12]. Ka Tartu Ülikooli bakalaureuseõppe informaatika õppekavas õpetatakse rekursiooni kursusel „Programmeerimine" (6 EAP) kursuse lõpus.

Rekursiooni temaatika käsitlemist alustatakse üldiselt selle mõiste defineerimisest ning selgitamisest. Sageli lisatakse teksti näitlikustamiseks ning paremaks mõistmiseks erinevaid programmikoode, visualiseeringuid ja reaalelulisi näiteid. Teema käsitlemine lõppeb sageli mingi programmeerimisülesandega. Enamasti on sellised näitlikustamised ja koodijupid autori poolt pikemalt lahti selgitatud ja programmis kommenteeritud.

Üldlevinud ülesanded, mida kasutatakse rekursiooni õpetamisel on Hanoi tornid, Fibonacci jada, faktoriaali arvutamine, fraktalite joonistamine jne. Neid ülesandeid on lihtne visualiseerida ja see aitab õppuril ülesannet paremini mõista. Visualiseerimine arendab intuitiivset arusaamist ning mõndade ülesannete puhul on võimalik visualiseerida programmi alamülesannete lahendusi. Järk-järgult ülesande lahendamine ning lahendusele lähemale jõudmine võimaldab kiiresti vigu tuvastada, neid parandada ja neist õppida [13].

4. „Programmeerimise alused II” rekursiooni temaatika

Käesolevas peatükis käsitletakse kursuse „Programmeerimise alused II” rekursiooni temaatika materjalide loomist ning lahenduste analüüsimist. Antud töös üldiselt ülesannetele õigeid lahendusi ei pakuta, põhjusel, et neid saaks edaspidi kursustel rakendada. Erinevad kursusel esinenud ülesanded on liigitatud alapeatükkideks. Igas alapeatükis käsitletakse üht teadmiste kontrolli liiki korraga, need liigid on:

- Moodle’i testid,
- programmeerimisülesanded,
- enesekontrolliülesanded,
- eksam

Kasutatud joonised on pärit Moodle’st, kuhu pääsevad ligi vaid kursusel osalejad ja korraldajad.

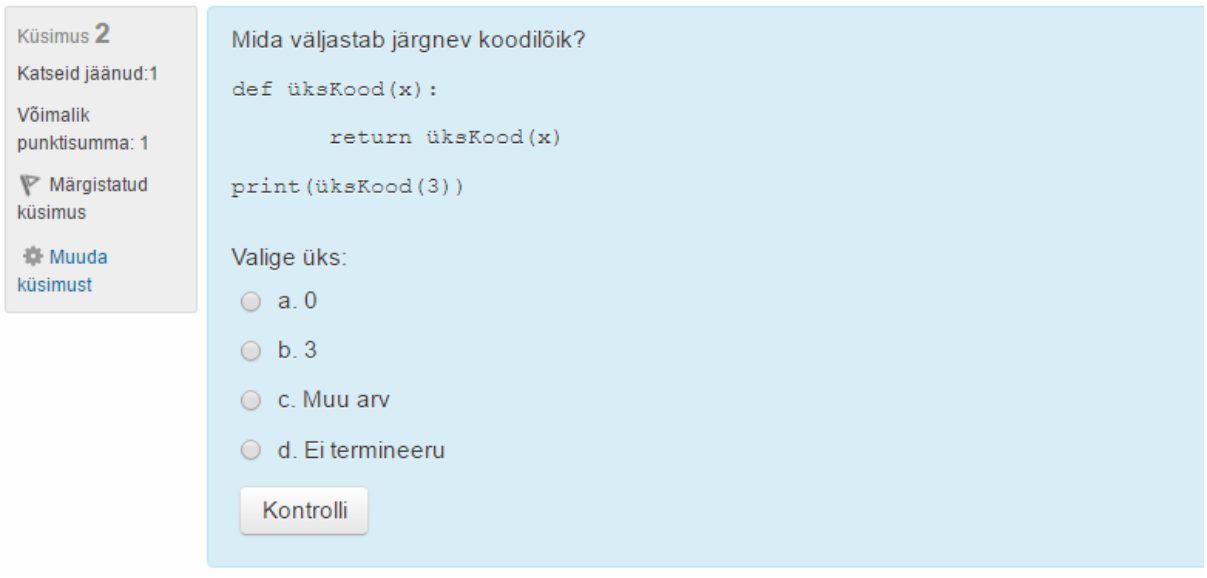
4.1 Moodle’i testid

Kursusel „Programmeerimise alused II” toimusid üle nädala testid, mida sooritati Moodle’i keskkonnas. Moodle on Tartu Ülikoolis veebipõhiste kursuste loomiseks kasutatav õpikeskkond [14].

Testid sisestati Moodle’i keskkonda käesoleva bakalaureusetöö autori poolt. Moodle’i testide eesmärk antud kursusel oli anda õppurile võimalus teadmisi kontrollida ja süvendada. Teisalt saavad ka õppejõud parema ülevaate õppurite järele jõudmisest ning teemade arusaadavusest. Testid koostati üldiselt koos kommentaaridega, enamuste õigete ning valede vastuste korral antakse tagasisidet. Küsimuste koostamisel lähtuti põhimõttest, et küsimustele oleks võimalik vastata ilma programmikoodi kirjutamata. Siiski oli mõndadele küsimustele kergem vastuseid leida programmikoodi Thonnysse kopeerides ja koodi juurde kirjutades. Testid olid kursusel osalejatele kohustuslikud.

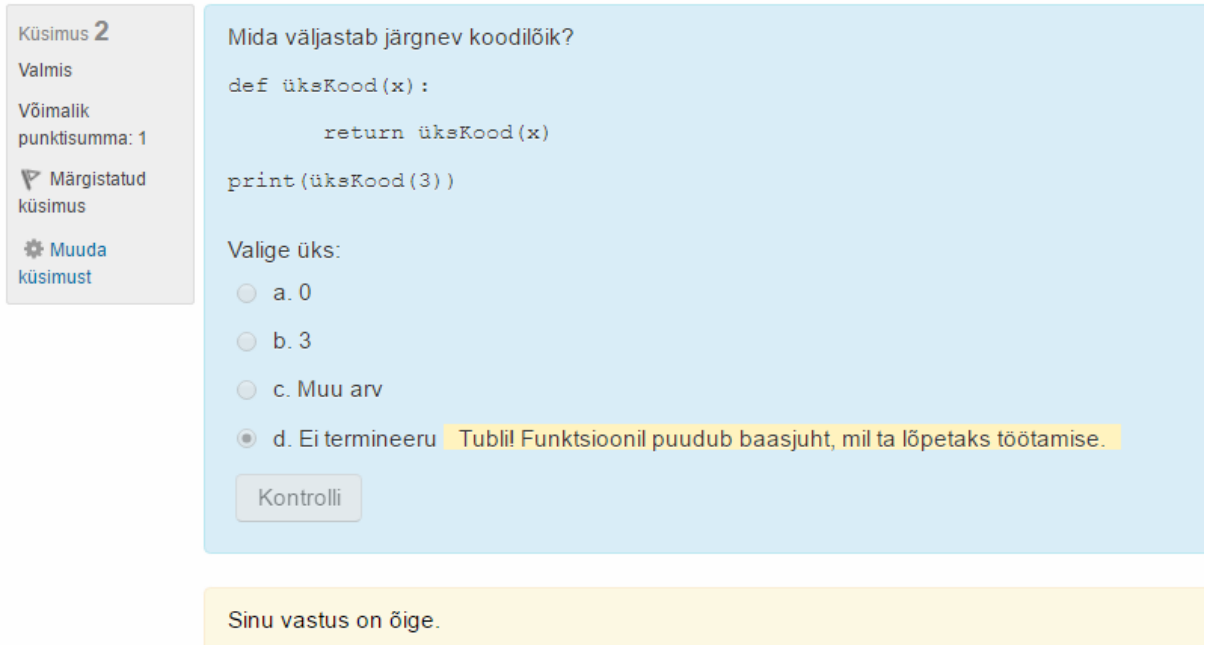
Rekursiooni teemal koostati kaks testi, mis toimusid teisel ning kolmandal kursuse perioodil. Testi läbimiseks nõutav tulemus oli vähemalt 90% kogupunktidest. Iga test andis aine punktides kokku 3 punkti, küsimusi oli testides vastavalt 4 ja 5. Testides oli võimalik saada vastavalt maksimaalselt 4 ja 5 punkti, lõpparvestuse jaoks teisendati need ümber 3-punkti skaalale. Teste võis sooritada lõpmatu arv kordi, testide puudus ajapiirang. Teise perioodi test sisaldas peamiselt küsimusi rekursiooni ja funktsiooni mõistmise kohta (vt Lisa I).

Küsimused olid loodud vastavalt, et kontrollida, kas õppur oskab programmikoodis tunda ära eelnevalt õpitud mõisteid ja definitsioone ning neid teadmisi rakendada. Näiteks teine küsimus testis kontrollis mittetermineeruva funktsiooni äratundmist (vt Joonis 1).



Joonis 1. II perioodi Moodle'i test, 2. küsimus vastuseta.

Õige vastuse ("Ei termineeru") valimisel antakse tagasiside, et vastus on õige ning selgitus, miks kood ei termineeru (vt Joonis 2).

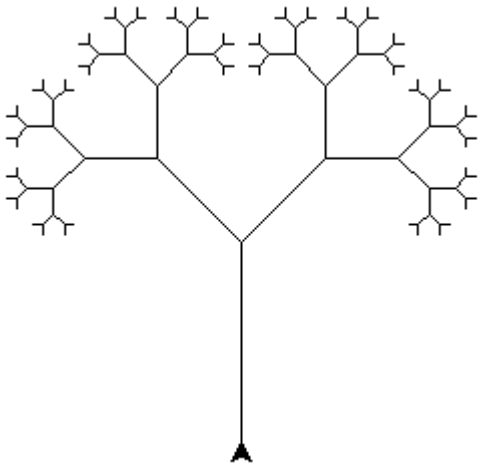


Joonis 2. II perioodi Moodle'i test, 2. küsimus vastuse ja tagasisidega.

Testi sooritati keskmiselt 1,76 korda. Kõige rohkem vastati valesti esimesele küsimusele, kokku 9 korda (vt Lisa I küsimus 1). Põhjuseks võib olla harva kasutatava jäägi leidmise

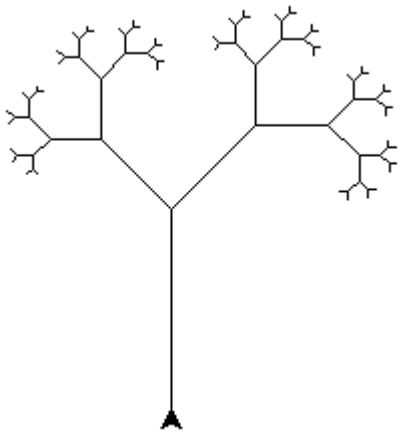
operaatori kasutamine koodis. Kasutatud operaator tagastab kahe arvu jagamisel saadava jäägi ning järgmine rekursiivne väljakutse kasutab seda jääki argumendina. Kõiki katseid arvestades oli keskmine tulemus 4, arvestades sooritanute kõrgemaid tulemusi, oli keskmine tulemus 4,8 punkti..

Kolmanda perioodi test oli võrreldes teise perioodi testiga keerukam (vt Lisa II). Küsimustes kasutatud programmikoodid olid pikemad ning vajasid rohkem keskendumist. Erinevuseks võib veel välja tuua küsimuse, kus sai õigeks vastuseks märkida mitu vastusevarianti, eelmise perioodi testis sai igas küsimuses märkida õigeks vaid ühe vastusevariandi (vt Lisa II, küsimus 1). Test sisaldas fraktaleid, täpsemalt puid, ja rekursiooni väljundi väljatrüki mõistmist (vt Lisa II). Testi sooritati keskmiselt 1,65 korda, mis on väiksem kui eelmise testi soorituste arv. Põhjuseks võib olla küsimuste arvu vahe testides, teises testis oli üks küsimus vähem, millest tulenevalt ka valesti vastatud küsimuste arvu tõenäosus oli väiksem. Kõige rohkem valesid vastuseid oli kolmandal küsimusel, kokku 6 (vt Lisa II küsimus 3). Selles küsimuses oli vaja etteantud koodijupis tühja lünga asenduseks valida vastusevariantidest õige valik, mille korral tekiks koodi väljundiks korrapärane puu, mis on kujutatud joonisel 3.



Joonis 3. Vastusevariandiga "puu(0.6 * pikkus)" saadav puu.

Vastatud kuuest valest vastusevariandist oli viie puhul vastatud vale variandina valik "puu(0.5 * pikkus)", mille korral konstrueeritakse küll programmi väljundina puu, kuid mitte korrapärane puu (vt Joonis 4).



Joonis 4. Vastusevariandiga “puu(0.5 * pikkus)” saadav puu.

Ilmselt jäi õppuritel küsimuse tekstis märkamata sõna „korrapärane”, mistõttu ka vale variant vastuseks valiti. Teisalt, selliseid ülesandeid, kus peaks lünga asemele uue koodiosa valima, pole varasemalt palju tehtud, mis võis olla õppuritele harjumatu. Kõiki vastuseid arvestades oli testi keskmine tulemus 3,36 punkti ja arvestades vaid sooritanute paremaid tulemusi 3,92 puntki.

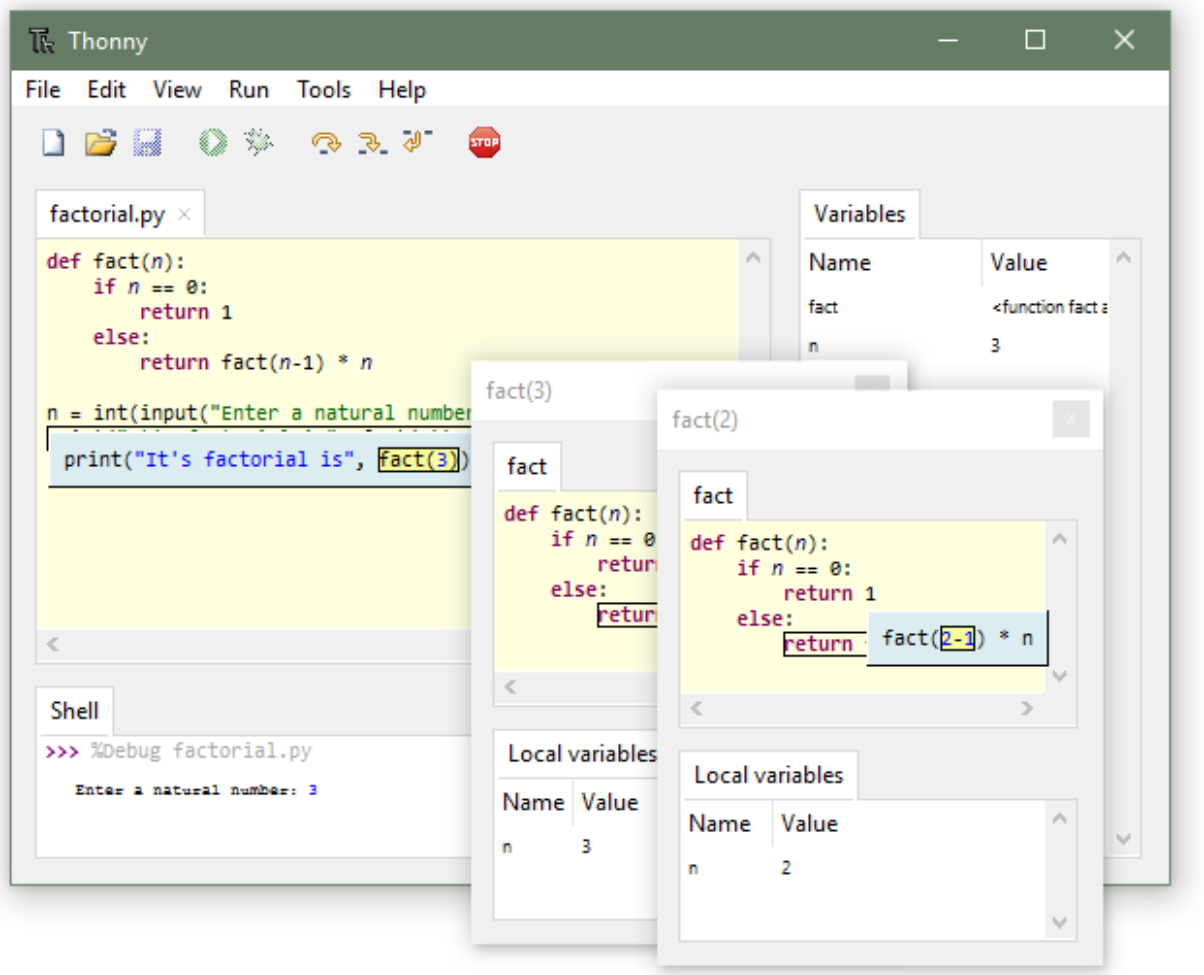
Kõik kursusele registreerunud 17 õppurit läbisid mõlemad rekursiooni testid edukalt. Kolmanda perioodi testi kohta toodi välja, oli ajakulukas ja üsna keeruline. Üldise tagasiside põhjal paistab, et vastava ülesehituse ning sisuga testid õppuritele sobisid ning olid huvipakkuvad, konkreetsemaid märkuseid rekursiooni testide kohta ei tehtud.

4.2 Programmeerimisülesanded

Käesolevas peatükis käsitletakse antud kursuse raames koostatud programmerimiseülesannetega seonduvat. Tutvustatakse ülesannete lahendamiseks kasutatud keskkonda Thonny, kirjeldatakse materjalide koostamise protsessi ja analüüsitakse lahendusi.

4.2.1 Programmeerimise keskkond Thonny

„Programmeerimise alused II” on Pythoni-põhine MOOC, mis tähendab, et kogu kursuse materjal on loodud programmeerimiskeeles Python. Kursusel kasutati programmeerimiseks vabavaralist keskkonda Thonny, mis on Aivar Annamaa loodud vahend Pythoni õppimiseks algajale. Thonny võimaldab programmi jooksutada sammhaaval ja jälgida, kuidas programm käitub, näiteks: mis väärtus omistatakse muutujale erinevatel sammudel (vt Joonis 5) [7].



Joonis 5. Kuvatõmmis Thonny programmist, kus koodi läbitakse sammhaaval (*debug*) [7].

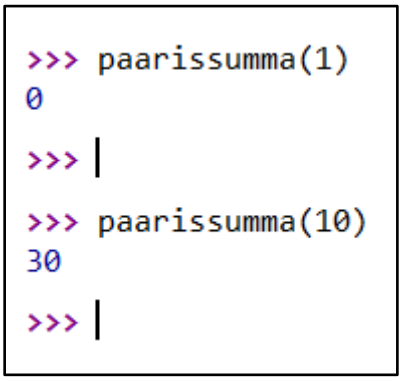
Thonny hõlbustab oma funktsionaalsustega programmeerimise protsessi nii õppuritele, kui ka abistab ülesannete kontrollijaid. Thonny kirjutab kasutaja tegevusest logifaili, mis on hiljem abiks õppejõududele ülesannete hindamisel ning analüüsimisel [15]. Logifaili on hiljem võimalik taasesitada ning täpselt jälgida õppuri programmi kirjutamise protsessi, sealhulgas näeb infot ajakulu, kopeerimiste, teksti sisestamise jpm kohta [15].

4.2.2 Programmeerimisülesannete loomine

Koduülesannete roll antud kursusel oli anda õppuritele võimalus etteantud ülesande püstituse põhjal iseseisvalt programmi kirjutada. Ühtlasi ka kontrollida, kas on teemadest aru saadud ja kas osatakse programme koostada. Oluliseks peeti vihjete ning koodi töötamise näidete andmist, et vältida ülesande püstitusest valesti aru saamist. Kodutööde lahendamiseks oli 2 nädalat aega, mille jooksul võis Moodle'i foorumis küsimusi esitada. Lahendusi võis Moodle'sse laadida soovitud arv kordi, hindamisel arvestati ainult viimast enne tähtaega esitatud varianti. Ülesanded olid koostatud põhimõttega, et lahenduseni jõudmine oleks

võimalik kasutades ainult antud kursusel etteantud materjale, kuulates loenguid ning käies praktikumis. Õppuritel oli hiljem võimalik oma Thonny logifailid Moodle'i keskkonda üles laadida, et kursuse korraldajad saaksid neid analüüsida ning järeldusi teha, seda võimalust kasutati vähe. Ülesandeid hinnati Moodle's automaatkontrollide põhjal ning hiljem sai õppejõud hinnet täpsustada. Hindamisprotsessis bakalaureusetöö autor ei osalenud.

Rekursiooni sisaldavaid kohustuslikke ülesandeid oli kokku 3, mida käsitleti teisel ja kolmandal perioodil. Lisaks oli neljandal perioodil 2 rekursiooni temaatika lisaülesannet. Teisel perioodil oli programmeerimisülesandeks „Paarissumma", mille kood pidi etteantud arvu a korral tagastama kõikide positiivsete paarisarvude summa, mis on väiksemad või võrdsed a-ga. Lisaks ülesande püstitusele oli ka ette antud näide vastava funktsiooni kasutamisest ning vihje selle lahendamiseks. Näitena olid ette antud juhud, kui argument on paaritu- või paarisarv, et näidata, milline on programmi väljund nendel juhtudel, näide on joonisel 6.



Joonis 6. Programmeerimisülesande „Paarissumma" näide Moodle's.

Vihjes toodi välja nipp lahendusele, mis sarnaneks faktoriaali arvutamise funktsioonile, mida on eelnevalt kursusel käsitletud. Veel oli vihjena ette antud, kuidas programmeerimisel vahet teha paaris- ja paaritul arvul (vt Lisa III).

Ülesanne andis kursuse arvestuses kokku 2 punkti. Valdav enamus, kogunisti 16 õppurit, sooritas ülesande maksimumpunktidele, 1 õppur sai tulemuseks 1,80 punkti. Antud juhul võeti 0,2 punkti maha, sest esitatud lahenduses puudus juht, kus etteantud argument oleks 1 ning seetõttu kood ei termineerunud. „Paarissumma" ülesandele logifaile ei esitatud. Moodlesse laaditud lahendustest ja tagasisidest selgus, et ülesanne oli sobiva raskusastmega ning kõigile jõukohane. Ülesande keskmine oli 1,99 punkti.

Kolmandal perioodil oli üheks programmeerimisülesandeks ülesanne „Rekursiooniga alla ja üles”. Ülesande täpne kirjeldus on joonisel 7, mis sisaldas ka väikest vihjet, eraldi vihjet sellel ülesandel antud ei olnud.

Koostada rekursiivne funktsioon `alla_üles`, mis saab argumendina ette ühe positiivse täisarvu. Kui argument on paaritu, siis väljastab funktsioon ekraanile kahanevalt positiivsed paaritud arvud, seejärel sõne "Baas!" ning siis kasvavalt paarisarvud alates 0 ja lõpetades vahetult enne argumenti.

Kui argument on paaris, siis väljastab funktsioon ekraanile kahanevalt positiivsed paarisarvud, seejärel sõne "Baas!" ning siis kasvavalt paaritud arvud lõpetades vahetult enne argumenti.

Ülesannet aitavad paremini mõista näited. (Vihjeks ka, et Moodle'i testis on sarnase ülesehitusega kood.)

Joonis 7. Programmeerimisülesande „Rekursiooniga alla ja üles” kirjeldus.

Ainsaks abistavaks vihjeks oli suunamine Moodle’i testini, kus oli taolise koodiga küsimus. Sarnaste ülesannete kasutamine sama kursuse erinevates teadmiste kontrolli liikides võiks motiveerida õppureid kõiki kursusel antud materjale kasutama. Sarnaste ülesannete puhul tekib äratundmisefekt ning õppurid hakkavad ülesannete vahel seoseid looma ning ei pea kogu lahendust algusest lõpuni ise mõtlema. Ka tavaelus kasutavad programmeerijad tihti juba valmis koodilõike ning täiendatakse neid vastavalt. Ülesandele olid näitena ette antud programmiväljundid nii paaris- kui ka paaritu arvu korral, näide on joonisel 8.

```
>>> alla_üles(7)
7
5
3
1
Baas!
0
2
4
6
>>>
```

```
>>> alla_üles(8)
8
6
4
2
Baas!
1
3
5
7
>>>
```

Joonis 8. Programmeerimisülesande „Rekursiooniga alla ja üles” näide.

Ülesanne andis kursuse arvestuses kokku 3 punkti. Kõik kursusel osalejad said ülesande eest maksimumpunktid. Enamasti oldi ülesannet lahendatud ühe if-i tingimuslause ja else-haruga.

Oli ka üksikuid lahendusi, kus oli rohkem tingimuslauseid kasutatud. 3 üliõpilast esitasid antud ülesandele ka logifaili. Logifailidest oli näha, et oldi malliks võetud kood Moodle'i testist ning sellest üritatud edasi arendada kodutöö tingimustele vastavat koodi.

Kolmanda perioodi teiseks rekursiooni ülesandeks oli „Pikima listi pikkus". Ülesandes oli kombineeritud mitmemõõtmeliste järjendite ja rekursiooni temaatika, et eelnevaid kursusel omandatud oskusi proovile panna. Lisaks ülesande püstitusele oli antud ka näide programmi töötamise kohta (vt Joonis 9).

Kirjuta funktsioon `max_pikkus`, mis võtab argumendiks listi, mille elementideks võivad olla täisarvud või listid, mille elementideks võivad olla täisarvud või listid, mille jne. Funktsioon peab tagastama pikima selles puntras leiduva listi pikkuse.

Näited:

- `max_pikkus([1,2,3])` peab tagastama *3*
- `max_pikkus([[[1,2,3]]])` peab samuti tagastama *3* (kõige sisemise listi pikkus on 3)
- `max_pikkus([[], [3,[4,5],[2,3,4,5,3,3], [7], 5, [1,2,3], [3,4]], [1,2,3,4,5]])` peab tagastama 7 (listil `[3,[4,5],[2,3,4,5,3,3], [7], 5, [1,2,3], [3,4]]` on 7 elementi)

Joonis 9. Programmeerimisülesande „Pikima listi pikkus" kirjeldus ja näide.

Ülesande püstituses olevatest näidetest joonistub hästi välja, et etteantud listi elementideks võivad olla ka teised listid listidest, täisarvud, tühjad listid jne. Et lihtsustada õppurite tööd ning harjutada neid kasutama sisseehitatud funktsioone, mis teevad sageli programmeerija tööd kergemaks, oli neile ette antud sellekohane vihje. Vihjes tutvustati Pythoni sisseehitatud funktsiooni `isinstance()`, mis aitab tõeväärtuste tagastamisega tuvastada, kas etteantud objekt on list (vt Joonis 10).

```
>>> isinstance(3, list)
False
>>> isinstance([3], list)
True
>>> isinstance([[3,4],2,5], list)
True
```

Joonis 10. Programmeerimisülesande „Pikima listi pikkus" vihje.

Ülesanne andis kursusel kokku 3 punkti ning kõik kursusel osalenud 17 õppurit said maksimumpunktid. Tagasisides toodi välja, et keeruline juht oli ülesande püstituses 2. näide, kus oli tegu kolmemõõtmelise järjendiga. Veel tõdeti, et raskust valmistas pikima listi pikkuse tagastamine, sest programm tagastas hoopis viimase listi pikkuse. Antud ülesande

kohta tõdesid tagasisides mitu õppurit, et ülesanne oli ajamahukas ning vajas palju süvenemist. Logifailidest selgus, et vihjet Moodle'i testi kohta ka kasutati ning kopeeriti kood Moodle'i testist ning üritati selle põhjal edasi arendada. Kõikides esitatud lahendustes oli seda funktsiooni kasutatud. Logifaile analüüsides selgus, et ülesandega oldi päris pikalt tegeldud ning erinevaid lahendusi katsetatud.

Lisaülesanded olid kursusel üpris populaarsed, mõlemale ülesandele laekus 5 lahendust, mis on peaaegu kolmandik kursusele registreerunutest. Mõlemale lisaülesandele laekunud lahendused olid samade inimeste poolt. Lisaülesanded olid lennuplaanide teemal, esimene oli „Lennuplaani otsing" ning teine „Lennuplaani koostamine". Ülesanded on püstitatud nii, et lahendus sisaldaks nii rekursiooni kui ka sõnastiku kasutust, mis on praktiline. Ülesanne „Lennuplaani otsing" peab tagastama tõeväärtuse, kas on võimalik lähtelinnast sihtlinna lennata (vt Joonis 11).

Koostada rekursioonil põhinev funktsioon `leidub_lennuplaan`, mis võtab argumentideks lähtelinna nime, sihtlinna nime ja lendude andmebaasi, ning tagastab tõeväärtuse vastavalt sellele, kas selle andmebaasi järgi on võimalik lennata lähtelinnast otse või ümberistumistega sihtlinna.

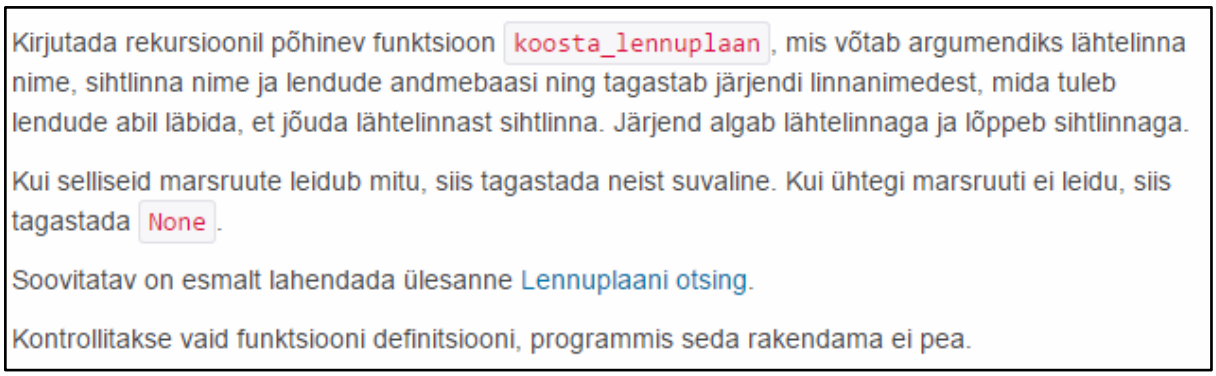
Lendude andmebaas on esitatud sõnastikuna, kus iga kirje võtmeks on linna nimi ja vastavaks väärtuseks hulk, mis sisaldab kõiki sellest linnast lähtuvate otselendude sihtkohti.

Kontrollitakse vaid funktsiooni definitsiooni, programmis seda rakendama ei pea.

Joonis 11. Programmeerimisülesande „Lennuplaani otsing" kirjeldus.

Üheks optimaalseks viisiks ülesande lahendamisel, oli teha abifunktsioon, et vältida tsükleid. Kahe funktsiooni koos kasutamist pole sel kursusel käsitletud, mis ilmselt vajas selle ülesande puhul rohkem süvenemist. Õppurite abistamiseks olid ette antud näited ja vihjed (vt Lisa IV). Ülesanne andis maksimaalselt ühe punkti, keskmine tulemus esitatud lahendustel oli 0,94 punkti. Neli lahendust viiest saavutasid maksimaalse tulemuse. Logifaile antud ülesandele ei esitatud.

Teine lisaülesanne oli „Lennuplaani koostamine", mis on sarnane esimesele, kuid veidi keerulisem. Esimeses tuli tagastada tõeväärtus, selles ülesandes tuleb tagastada järjend linnanimedest, mida tuleb sihtkohta jõudmiseks läbida, ülesande püstitus on joonisel 12.



Joonis 12. Programmeerimisülesande „Lennuplaani koostamine" kirjeldus.

Ülesandes oli ette antud ka näide ja vihjed (vt Lisa V). Näide sisaldas andmebaasi väljatrükki ning funktsiooni kasutuse näidet. Sarnaselt eelmisele lisaülesandele, oli ka selle eest võimalik maksimaalselt saada ühe punkti, keskmine tulemus esitatud lahenduste seas oli 0,92 punkti. Esitatute hulgas oli 3 maksimaalse tulemusega lahendust ning kahe teise lahenduse tulemus oli 0,8 punkti. Logifaile ülesandele ei esitatud. Lisaülesannete kohta tõdeti tagasisides nii lahendajate kui ka lahendusi mitte esitanud õppurite poolt, et ülesanded olid kõrge raskusastmega ja vajasid palju aega.

Tudengite tagasisidest selgus, et programmeerimisülesannete lahendamisel oli vihjetest ja näidetest palju abi. Ülesanded küll sooritati väga headele tulemustele, kuid sellest ei saa järeldada, et ülesanded oleksid liiga lihtsad olnud. Vaadates õppurite teiste temaatikate ülesannete tulemusi samal kursusel, on keskmised tulemused samaväärsed. Veel tõdeti tagasisides, et rekursioon on üsna keeruline teema ning ülesannete lahendamine võttis oodatust rohkem aega.

4.3 Enesekontrolli küsimused

Lisaks eelnevalt käsitletud materjalidele, oli kursusel veebipõhine õpik, mis on kättesaadav Courses.cs.ut.ee keskkonnas [16]. Õpik on jaotatud peatükkideks kursusel käsitletavate teemade järgi. Rekursiooni temaatikat käsitletakse õpikus pärast kordamise ning kahemõõtmeliste järjendite peatükke. Seal on defineeritud mõisteid, analüüsitud programmikoode ning selgitatud programmide käitumist. Tegemist pole tavalise õpikuga, vaid interaktiivsega, kus saab end kontrollida ja selle käigus õppida. Teksti vahel on kontrollküsimused, mis annavad tagasisidet (vt Joonis 13).

Joonis 13. Enesekontrolli ülesanne Courses.cs.ut keskkonnas [16].

Neid võib lahendada mistahes arv kordi, see info aine vastutajateni ei jõua. Enesekontrolli küsimused on õppurite endi jaoks, hindamisel neid kuidagi arvesse ei võeta. Nende küsimuste eripäraks on konkreetse tagasiside andmine erinevate vastuste puhul, mis on välja toodud tabelis 1.

Tabel 1. Enesekontrolli küsimuse „süt" tagasiside erinevate vastuste korral.

Vastus	Kommentaar
4	Õige, programm väljastab etteantud arvude suurima ühisteguri.
8	Vale, jääk 20 jagamisel 12-ga on küll 8, aga seda ei väljastata, sest programm töötab edasi.
20	Vale, esimesel sammul a == 20, kuid b ei ole siis veel 0 ning programm sinna ei jõua.
Muu arv	Vale, õige arv on ülevalpool olemas.
Ei termineeru	Vale, termineerub kenasti.

Rekursiooni temaatikal oli antud kursusel 1 enesekontrolli küsimus. Küsimuses olnud programmikood oli kahe arvu suurima ühisteguri ehk süt-i leidmise algoritm.

Õppurite tagasisidest selgus, et need olid õppimisel abiks ning tegid õpiku lugemise huvitavamaks. Lisati ka, et neid oleks võinud olla rohkem.

4.4 Näidiseksam ja eksam

E-kursus „Programmeerimise alused II" lõppes eksamiga. Eksamil oli rekursiooni temaatikal 2 ülesannet. Eksamiks valmistumiseks oli Moodle's näidiseksam, mis andis aimu eksamile tulevatest ülesandetüüpidest ning raskusastmest. Moodle'st olid kättesaadavad näidiseksami lahendused ning õppevideod nende lahendamisest. Õppevideode tegemisel bakalaureusetöö

autor ei osalenud. Eksami sooritamiseks oli ette nähtud 180 minutit, mille jooksul pidi lahendama nii paber- kui ka arvutiosa ülesande. Mõlemad osad andsid 20 punkti, eksamil oli kokku võimalik saada 40 punkti.

Paberosa ülesande lahendamisel oli keelatud kasutada arvutit, materjale ning suhelda kaasõppuritega. Positiivse hinde jaoks oli vaja saada vähemalt 10 punkti. Kõrgelt hinnati õigeid vastuseid ja selgitusi, samas ka selgitusi, mis näitavad mõistlikku mitmevahelolekut. Näidiseksami paberosa ülesanne oli koodi analüüsimise ning mõistmise kohta, kõikidele vastustele nõuti põhjendusi (vt Lisa VI paberosa ülesanne). Sarnase põhimõttega koodi oldi varasemalt loengus läbi tehtud. Eksamil oli näidiseksami ülesandega sarnane paberosa ülesanne: oli ette antud kood, mis võtab kaks argumenti ning programmi töö käigus tehakse nende arvudega elementaartehteid, igal sammul trükitakse üks neist argumentidest ekraanile. Õppurid pidid ülesande lahenduseks kirjutama täpse programmi väljundi.

Eksami arvutiosa ülesanne lahendati arvutits, kasutada võis materjale, kuid suhtlemine oli keelatud. Töö esitati Moodle'i keskkonnas, positiivse hinde jaoks oli vaja vähemalt 10 punkti. Ülesandeks oli koostada programm etteantud kirjelduse ja koodi töötamise näite põhjal. Üheks näidisülesandeks oli „Paarissumma" ülesanne, mis oli kursusel varasemalt ka koduseks programmeerimisülesandeks (vt Joonis 6). Teiseks näidisülesandeks oli ülesanne, mis prindib igal sammul etteantud argumendist kolme võrra väiksema arvu ning võtab uueks argumendiks saadud arvu (vt Lisa VI arvutiosa ülesanne). Programm pidi tagastama esimese arvu, milleni jõuti, mis ei ole positiivne. Eksamil olid näidiseksamile sarnase raskusastme ning püstitusega ülesanded.

Eksamiülesannete logifaile ei küsitud ning mõlemast osast saavutatud tulemused on Moodle'i keskkonda lisatud summeeritud kujul, seega ei saa tuvastada, kui palju teatud ülesannete eest punkte saadi. Moodle'st on näha, et paberosa eest on 4 õppurit ning arvutiosa eest 5 õppurit saanud maksimumtulemuse, millest järeldub, et vähemalt 5 inimest on ka rekursiooni ülesanded sooritanud maksimaalsele tulemusele. Parimad tulemused, mis eksamil saavutati on kahe õppuri poolt saavutatud 40 ja 39 punkti. Kõik osalejad läbisid kursuse edukalt.

5. Kokkuvõte

Käesoleva bakalaureusetöö eesmärk oli luua MOOCile „Programmeerimise alused II" rekursiooni temaatika ülesandeid ja küsimusi. Koostatud materjali analüüsiti õppurite tulemuste ning tagasiside põhjal. Vaatluse alla võeti kursus, mis toimus ajavahemikus 20. oktoober – 15. detsember 2016. Materjalid loodi suuresti koostöös koos teiste kursuse korraldajate ning abistajatega, et tagada materjalide kvaliteet.

Bakalaureusetöö autor koostas kursuse teisel ja kolmandal perioodil ühe Moodle'i testi. Teise perioodi testile laekus enim valesid vastuseid ülesandele, mille kood sisaldas jäägi leidmise operaatorit. Töös toodi välja, et see ei olnud õppuritele tuttav operaator ning seetõttu ka valesti vastati. Kolmanda perioodi testis oli kõige problemaatilisemaks ülesandeks koodijupi täiendamise ülesanne. Lahendusena leiti, et ilmselt jäi paljudel märkamata, et küsitud oli korrapärast puud, mistõttu ka paljud esmalt sellele küsimusele valesti vastasid.

Töö raames koostati ka programmeerimisülesandeid, töö autor osales ülesannete loomisel ja täiendamisel. Kokku koostati 5 programmeerimisülesannet, millest 3 olid kohustuslikud ning 2 lisaülesanded. Teise perioodi ülesanne oli kõigile jõukohane ning tulemuste keskmine oli kõrge. Kolmandal perioodil käsitletava ülesande püstitus oli sarnane sama nädala Moodle'i testis oleva ülesandega. Ülesandes antud vihjele vaatamata, pidid õppurid siiski ise koodi piisavalt mõistma, et uue püstitusega ülesannet lahendada. Kõige keerukamaks rekursiooni temaatika ülesandeks võib pidada ülesannet „Pikima listi pikkus", mille kohta mitmed õppurid tõdesid, et ülesanne võttis päris palju aega ja vajas korralikult nuputamist. Samas lahendasid kõik tudengid selle lõpuks maksimumpunktidele. Rekursiooni temaatikal olid ka 2 lisaülesannet, mõlemale laekus 5 lahendust, mis olid kõik hästi lahendatud. Tagasiside põhjal paistis, et lisaülesanded ei olnud kellelegi liiga kerged ehk võib järeldada, et lisaülesanded täitsid oma eesmärki.

Õppurite enesekontrolliks loodi töö käigus Courses.cs.ut.ee keskkonda üks ülesanne, mis sai hulganisti head tagasisidet ning tõdeti, et neid võiks olla palju rohkem.

Töö autor osales ka eksami ja selle näidisülesannete välja töötamisel. Mõlemad sisaldasid nii paber- kui ka arvutiosa ülesandeid. Kokku koostati ülesandeid näidiseksami ning kahe korralise eksami jaoks kokku 10, lõplikult läks neist kasutusse 7. Eksamiülesandeid detailsemalt analüüsida ei olnud võimalik. Üldistest tulemusest selgub, et kõik kursusele registreerunud läbisid kursuse edukalt.

Autor loodab, et antud töö on tulevikus abiks MOOCide rekursiooni temaatika ülesannete, küsimuste ning ka tekstiliste materjalide loomisel.

6. Viidatud kirjandus

[1] By The Numbers: MOOCS in 2016. https://www.class-central.com/report/mooc-stats-2016/ (8.05.2017)

[2] Quality education for everyone, everywhere. https://www.edx.org/about-us (6.05.2017)

[3] Study of MOOCs offers insights into online learner engagement and behavior. http://blog.edx.org/study-moocs-offers-insights-online-learner-engagement-behavior?track=blog (13.02.2017)

[4] Marques, J., McGuire, R. (2013). What is a Massive Open Online Course Anyway? MN+R Attempts a Definition. http://moocnewsandreviews.com/what-is-a-massive-open-online-course-anyway-attempting-definition/

[5] The 50 Most Popular MOOCs of All Time. http://www.onlinecoursereport.com/the-50-most-popular-moocs-of-all-time/ (27.04.2017)

[6] Programmeerimisest maalähedaselt, kevad 2015/16. https://courses.cs.ut.ee/2016/progmaa/spring/Main/Maalahealusedvordlus/ (6.05.2017)

[7] Thonny ametlik koduleht. http://thonny.org/ (25.05.2017)

[8] Programmeerimise algkursus. https://courses.cs.ut.ee/MTAT.03.100/2012_fall/uploads/opik/12_rekursioon.html (8.05.2017)

[9] Zelle JM. (2002). Python programming: an introduction to computer science. Franklin, Beedle & Associates, Inc.

[10] Fractal art. http://www.arthistory.net/fractal-art/ (11.04.2017)

[11] Burns A. M. (2005). Recursion in Nature, Mathematics and Art URL http://symmetry-us.com/Journals/bridges2005/burns/index.html

[12] Kiho, J. Java programmeerimise aabits. Tartu: Tartu Ülikooli Arvutiteaduse instituut. 2012.

[13] Dann W, Cooper S, Pausch R. Using visualization to teach novices recursion. ACM SIGCSE Bulletin. 2001 Sep 1;33(3):109-12.

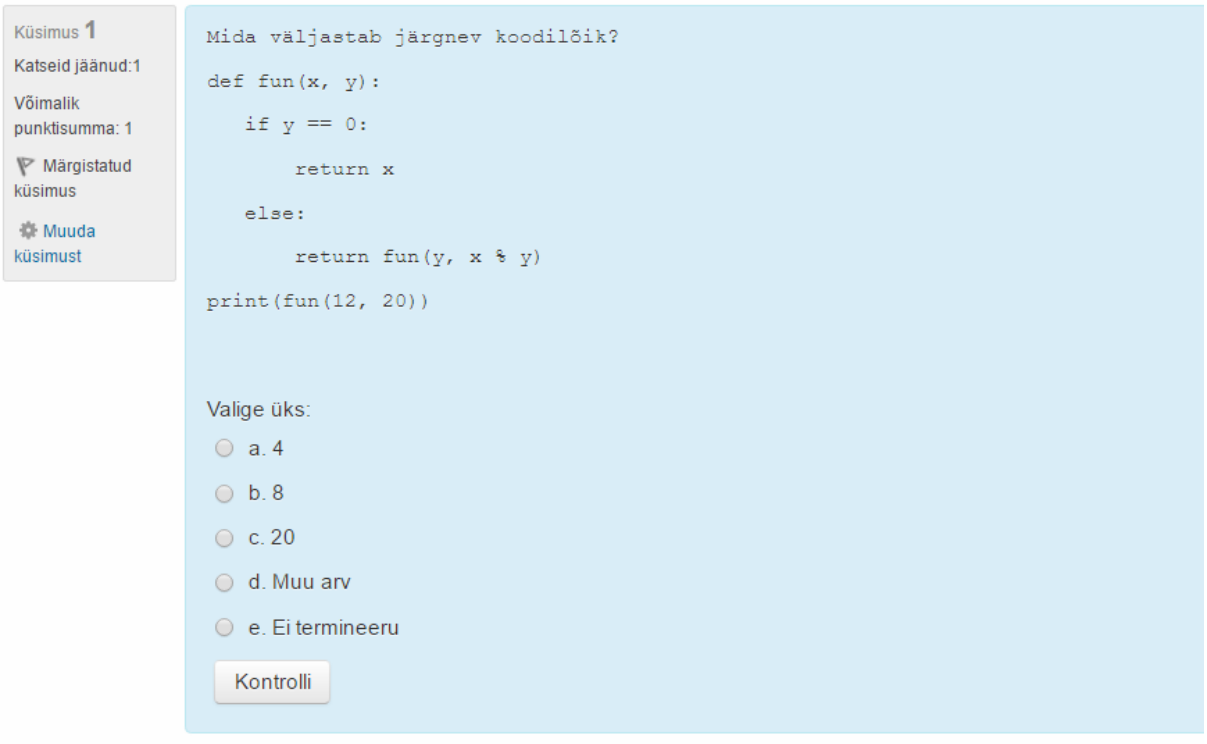
[14] Moodle. https://sisu.ut.ee/juhendid/moodle (8.05.2017)

[15] Pedel, K. E-kursuse "Programmeerimise alused" logifailide analüüs. TÜ arvutiteaduse instituudi bakalaureusetöö. 2016.

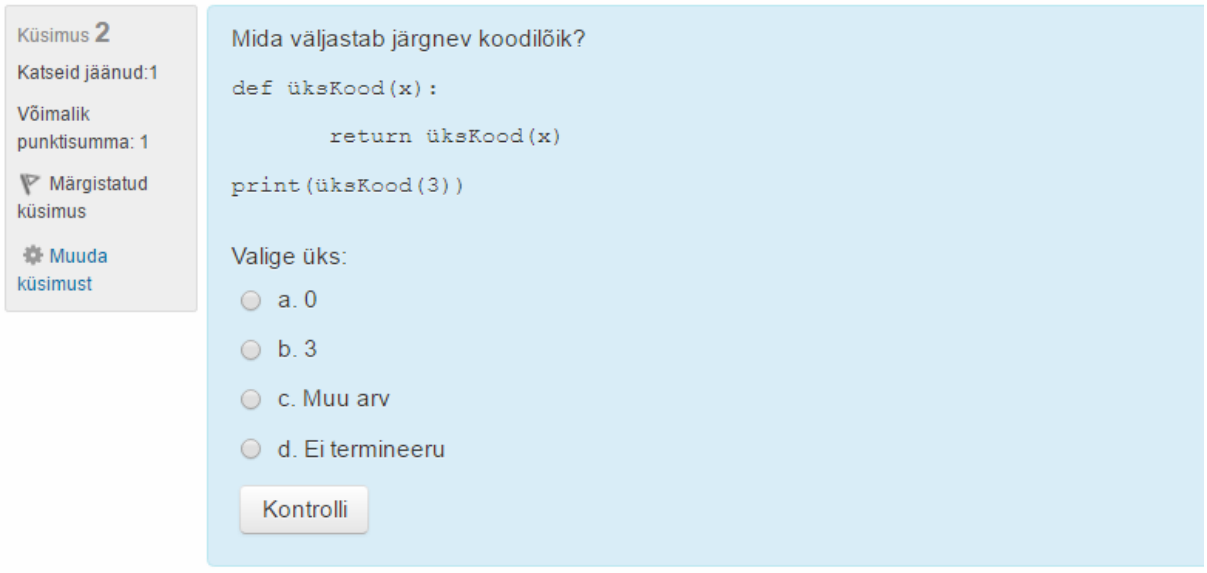
[16] Programmeerimise alused II. https://courses.cs.ut.ee/2016/alused2/fall/Main/PART2A (28.04.2017

Lisad

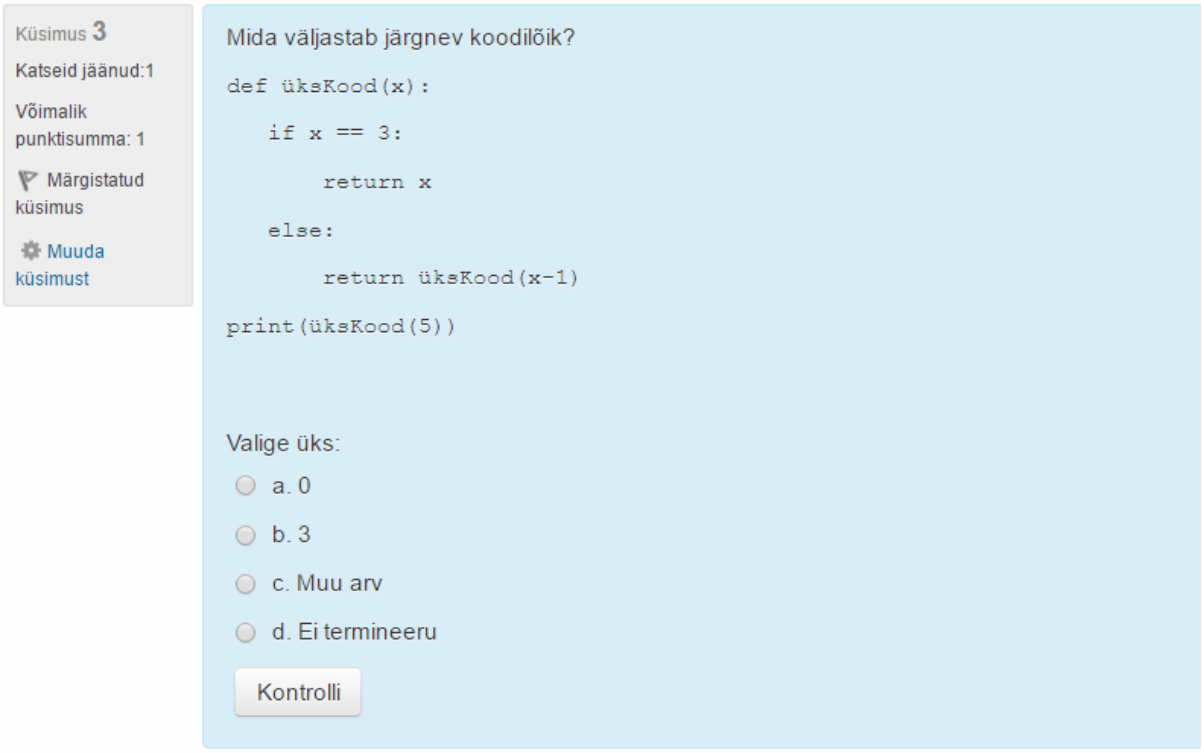
I 2. perioodi rekursiooni Moodle'i test



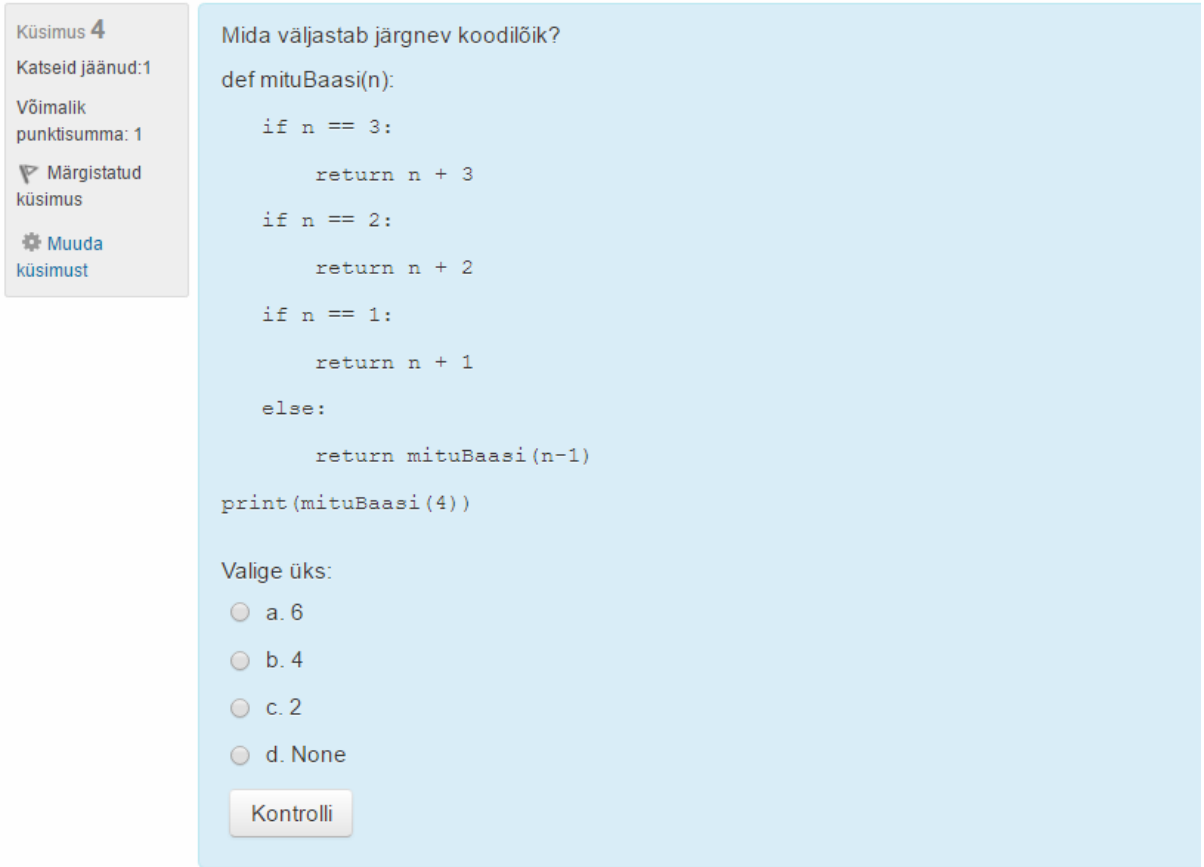
Küsimus 1.



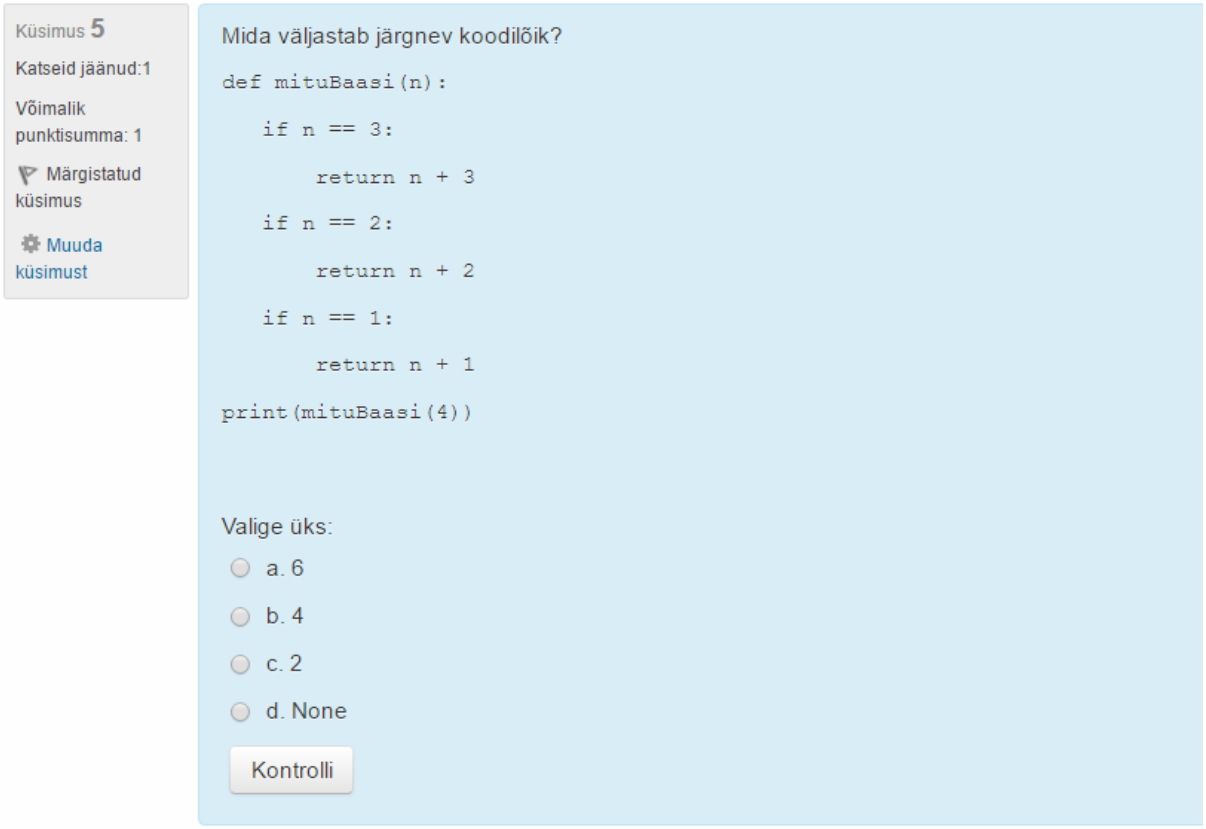
Küsimus 2.



Küsimus 3.

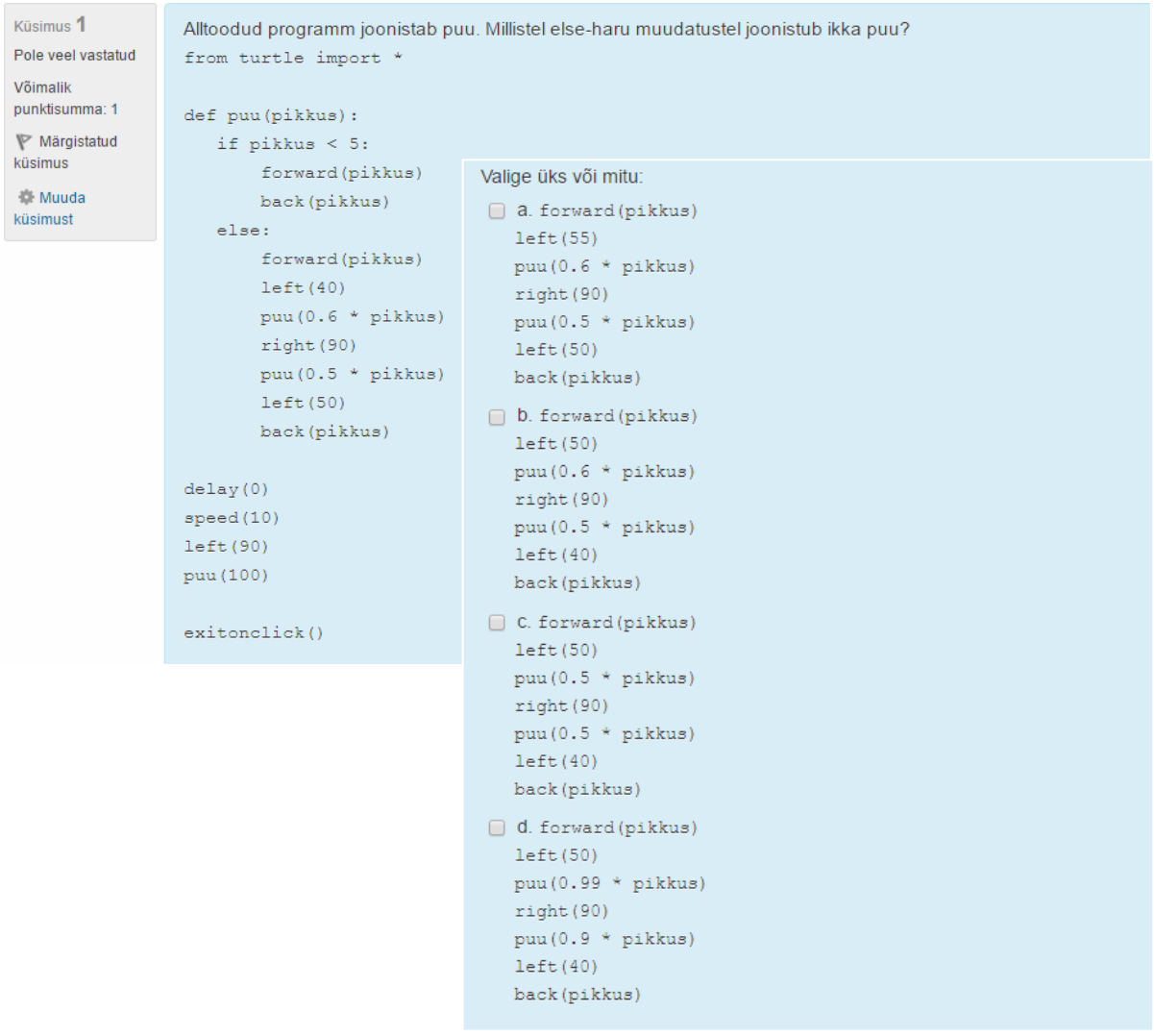


Küsimus 4.


Küsimus 5
Katseid jäänud:1
Võimalik punktisumma: 1
Märgistatud küsimus
Muuda küsimust
Mida väljastab järgnev koodilõik?
def mituBaasi(n):
if n == 3:
return n + 3
if n == 2:
return n + 2
if n == 1:
return n + 1
print(mituBaasi(4))
Valige üks:
a. 6
b. 4
c. 2
d. None
Kontrolli

Küsimus 5.

II 3. perioodi rekursiooni test



Küsimus 1 | Pole veel vastatud | Võimalik punktisumma: 1 | Märgistatud küsimus | Muuda küsimust

Alltoodud programm joonistab puu. Millistel else-haru muudatustel joonistub ikka puu?

```
from turtle import *

def puu(pikkus):
    if pikkus < 5:
        forward(pikkus)
        back(pikkus)
    else:
        forward(pikkus)
        left(40)
        puu(0.6 * pikkus)
        right(90)
        puu(0.5 * pikkus)
        left(50)
        back(pikkus)

delay(0)
speed(10)
left(90)
puu(100)

exitonclick()
```

Valige üks või mitu:

a.
```
forward(pikkus)
left(55)
puu(0.6 * pikkus)
right(90)
puu(0.5 * pikkus)
left(50)
back(pikkus)
```

b.
```
forward(pikkus)
left(50)
puu(0.6 * pikkus)
right(90)
puu(0.5 * pikkus)
left(40)
back(pikkus)
```

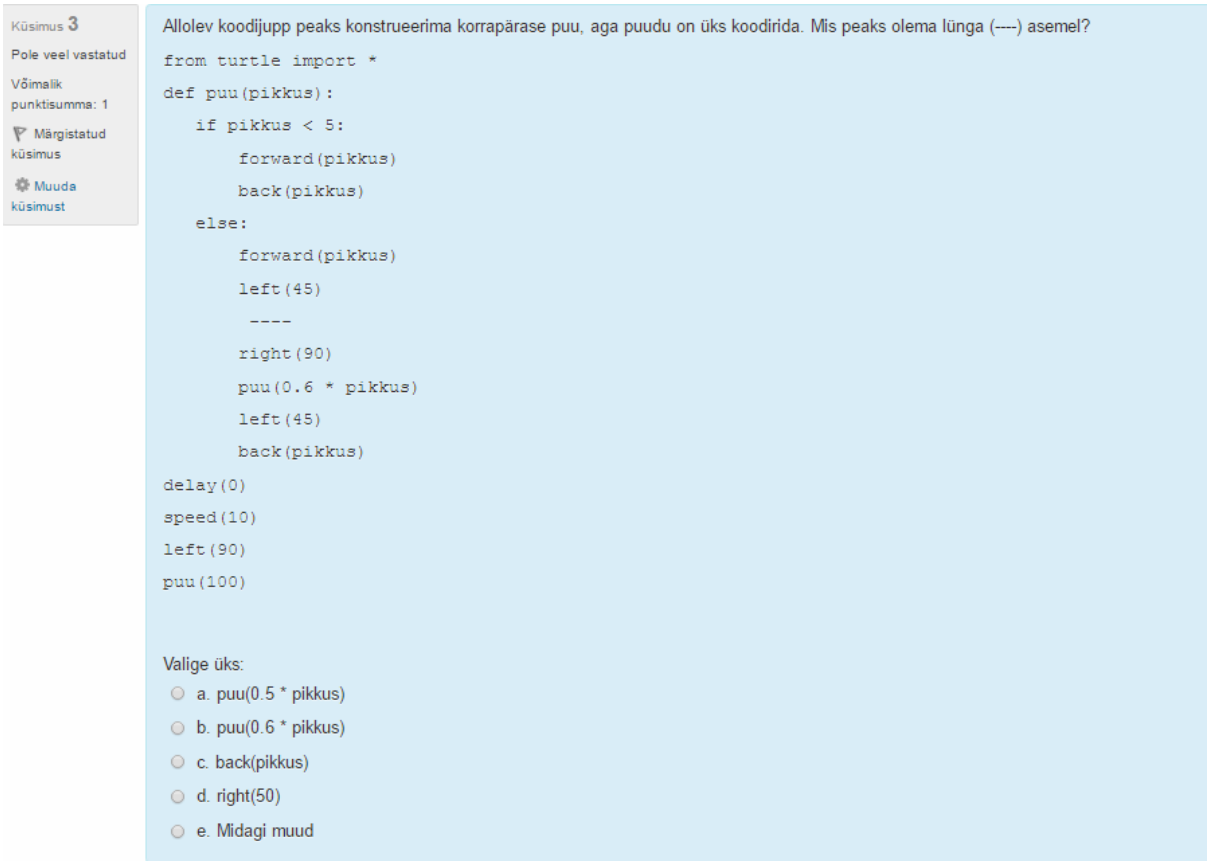
c.
```
forward(pikkus)
left(50)
puu(0.5 * pikkus)
right(90)
puu(0.5 * pikkus)
left(40)
back(pikkus)
```

d.
```
forward(pikkus)
left(50)
puu(0.99 * pikkus)
right(90)
puu(0.9 * pikkus)
left(40)
back(pikkus)
```

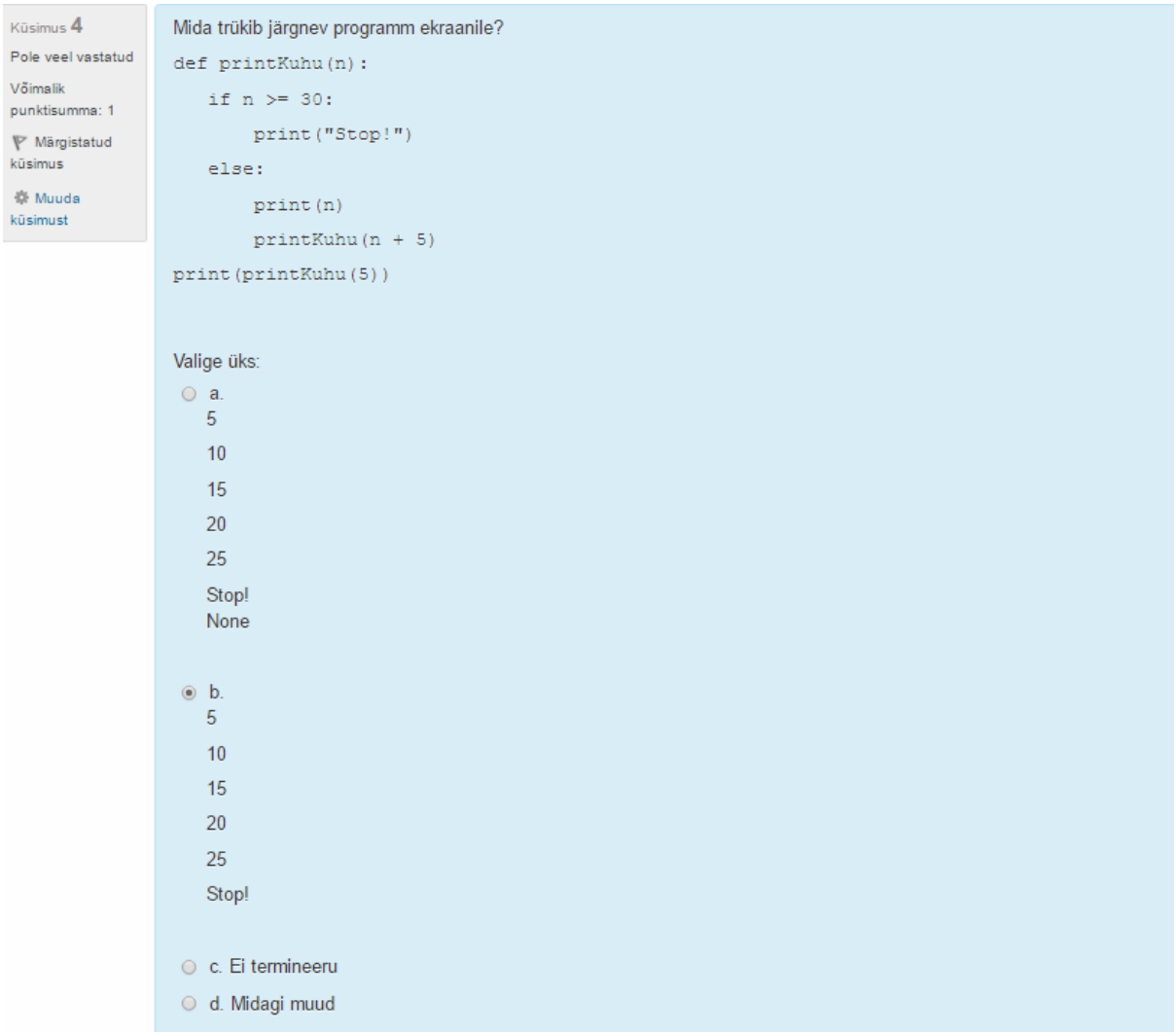
Küsimus 1.


Küsimus 2
Pole veel vastatud
Võimalik punktisumma: 1
Märgistatud küsimus
Muuda küsimust
Mida trükib järgnev programm ekraanile?
def printKuhu(n):
if n <= 0:
print("Stop!")
else:
print(n - 1)
printKuhu(n - 1)
print(n)
print(printKuhu(2))
Valige üks:
a.
1
0
Stop!
1
2
None
b.
1
0
Stop!
1
2
c. Ei termineeru
d. Midagi muud

Küsimus 2.


Küsimus 3
Pole veel vastatud
Võimalik punktisumma: 1
Märgistatud küsimus
Muuda küsimust
Allolev koodijupp peaks konstrueerima korrapärase puu, aga puudu on üks koodirida. Mis peaks olema lünga (----) asemel?
from turtle import *
def puu(pikkus):
if pikkus < 5:
forward(pikkus)
back(pikkus)
else:
forward(pikkus)
left(45)
----
right(90)
puu(0.6 * pikkus)
left(45)
back(pikkus)
delay(0)
speed(10)
left(90)
puu(100)
Valige üks:
a. puu(0.5 * pikkus)
b. puu(0.6 * pikkus)
c. back(pikkus)
d. right(50)
e. Midagi muud

Küsimus 3.


Küsimus 4
Pole veel vastatud
Võimalik punktisumma: 1
Märgistatud küsimus
Muuda küsimust
Mida trükib järgnev programm ekraanile?
def printKuhu(n):
if n >= 30:
print("Stop!")
else:
print(n)
printKuhu(n + 5)
print(printKuhu(5))
Valige üks:
a.
5
10
15
20
25
Stop!
None
b.
5
10
15
20
25
Stop!
c. Ei termineeru
d. Midagi muud

Küsimus 4.

III Programmeerimisülesande „Paarissumma” vihje

Rekursiivse funktsiooni korral on oluline hargnemine. Vähemalt üks haru peab olema ilma rekursiivse väljakutseta (rekursiooni baas). Rekursiivse väljakutsega harus peab funktsioon väljakutsutama väiksema argumendiga - peab toimuma liikumine baasi poole.

Õige lahenduseni võib jõuda mitut moodi. Näiteks võime esialgu püüda leida kõigi positiivsete arvude summa, mis on väiksemad või võrdsed a-ga. See funktsioon oleks üsna sarnane faktoriaali arvutamise funktsioonile.

Antud juhul pole aga kõiki vaja. Võime siis modifitseerida nii, et baasi poole liigutakse a - 1 asemel a - 2. Erinevused on aga paaris- ja paaritute arvude puhul. Nende vahel saab vahet teha a % 2 == 0 abil.

Abiks võib olla tähelepanek, et paarituarvulise argumendi puhul on paarissumma sama, mis eelneva paarisarvu puhul: paarissumma(5) on sama, mis paarissumma(4).

IV Programmeerimisülesande „Lennuplaani otsing” näide ja vihjed

```
>>> lennud = {"Pariis" : {"London", "Berliin", "Nice"}, "London" : {"Berliin", "Pariis"},
"Berliin" : {"London", "Pariis", "Tallinn"}, "Tallinn" : {"Berliin"}}
>>> leidub_lennuplaan("Tallinn", "Nice", lennud)
True
>>>
```

Küsimusele vastamine on lihtne siis, kui lähte- ja sihtlinn on samad. Kõik keerulisemad küsimused saab taandada eelnevale.

Tsüklite vältimiseks võib funktsiooni põhiloogika delegeerida teisele funktsioonile, mis võtab lisaks argumendi, mille abil on võimalik meeles pidada, mis linnad on juba külastatud.

V Programmeerimisülesande „Lennuplaani koostamine” näide ja vihjed

```
>>> lennud = {"Pariis" : {"London", "Berliin", "Nice"}, "London" : {"Berliin", "Pariis"},
"Berliin" : {"London", "Pariis", "Tallinn"}, "Tallinn" : {"Berliin"}}
>>> koosta_lennuplaan("Tallinn", "Nice", lennud)
['Tallinn', 'Berliin', 'Pariis', 'Nice']
>>> koosta_lennuplaan("Tallinn", "Nice", lennud)
['Tallinn', 'Berliin', 'London', 'Pariis', 'Nice']
>>>
```

Vahemaandumised saab meelde jätta "tagasiteel" pärast sihtkoha leidmist.

Tsüklite vältimiseks võib funktsiooni põhiloogika delegeerida teisele funktsioonile, mis võtab lisaks argumendi, mille abil on võimalik meeles pidada, mis linnad on juba külastatud.

VI Näidiseksami rekursiooni ülesanded

Mida trükib ekraanile järgmine programmilõik?

```
def rekFun(x, y):
    print(x)
    if y < 0:
        return x
    else:
        return rekFun(x + 2, y - 1) + rekFun(x + 4, y - 1)
print(rekFun(-2, 1))
```

Paberosa ülesanne.

Koostada rekursiivne funktsioon, mis saab argumendina ühe positiivse täisarvu. Funktsioon peab (igal väljakutsel) ekraanile väljastama argumendi. Rekursiivselt peab korraldama, et igal järgmisel väljakutsel on argument eelmisest 3 võrra väiksem ja argument oleks positiivne. Lõpuks peab funktsioon tagastama esimese sellise arvu, mis pole positiivne.

Kasutamise näited

```
print("Tagastatakse: " + str(fun(11)))
```

11
8
5
2
Tagastatakse: -1

```
print("Tagastatakse: " + str(fun(6)))
```

6
3
Tagastatakse: 0

```
print("Tagastatakse: " + str(fun(2)))
```

2
Tagastatakse: -1

```
print("Tagastatakse: " + str(fun(-3)))
```

Tagastatakse: -3

Arvutiosa ülesanne.

VII Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, **Kirsti Tagam**,
(*autori nimi*)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose
E-kursuse „Programmeerimise alused“ rekursiooni temaatika küsimuste ja ülesannete loomine,
(*lõputöö pealkiri*)

mille juhendaja on Eno Tõnisson,
(*juhendaja nimi*)

1.1. reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

1.2. üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.
3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, **11.05.2017**