TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

**Aleksander Ontin**

# Tartu Ülikooli kursuse “Programmeerimise alused II” ülesannete ja murelahendajate loomine

**Bakalaureusetöö (9 EAP)**

Juhendaja: Heidi Meier

Tartu 2024

# Tartu Ülikooli kursuse “Programmeerimise alused II” ülesannete ja murelahendajate loomine

**Lühikokkuvõte:**

Tartu Ülikooli kursus "Programmeerimise alused II" on programmeerimise õppeaine, mis toimub kevadsemestril ning annab üliõpilastele süvendatud alusteadmised programmide koostamisest, testimisest ja silumisest. Selle bakalaureusetöö eesmärk on luua sellele kursusele uued ülesanded ja neile vastavad murelahendajad. Kursuse käigus kasutatakse programmeerimise ülesandeid, mis on jagatud koduülesanneteks ja praktikumi ülesanneteks. Murelahendaja on järjestikuste küsimuste kogum koos vihjete ja koodinäidetega, mis on mõeldud selleks, et aidata üliõpilastel lahendada erinevaid probleemseid kohti ülesannetes. Töö raames koostati kokku 39 ülesannet ja nende jaoks loodi 26 murelahendajat. Ülesannete ja murelahendajate hindamiseks kasutati tagasiside küsimustikku. Tagasiside küsimustiku tulemuse põhjal suhtusid paljud kursuse üliõpilased positiivselt ülesannetesse ja osadesse murelahendajatesse.

**Võtmesõnad:**

murelahendaja, programmeerimise õpetamine, programmeerimise ülesanded

**CERCS:** P175 Informaatika, süsteemiteooria, S270 Pedagoogika ja didaktika

# Creating of Exercises and Troubleshooters for the Course “Introduction to Programming II” at the University of Tartu

**Abstract:**

The University of Tartu's “Introduction to Programming II” course is a programming subject that takes place in the spring semester and provides students with in-depth basic knowledge of programming, testing and debugging. The aim of this bachelor thesis is to create new exercises and corresponding troubleshooters for this course. During the course, programming exercises will be used, which will be divided into homework and lab exercises. The troubleshooter is a set of sequential questions with hints and code examples designed to help students solve different problem areas in the exercises. A total of 39 exercises were designed and 26 troubleshooters were created for them. A feedback questionnaire was conducted and used to evaluate the exercises and the troubleshooters. Based on the results of the feedback questionnaire, many of the students on the course had a positive attitude towards the exercises and some of the troubleshooters.

**Keywords:**

troubleshooter, teaching programming, programming exercises

**CERCS:** P175 Informatics, systems theory, S270 Pedagogy and didactics

## Sisukord

## Sissejuhatus

Kursus “Programmeerimise alused II” (MTAT.03.256) on Tartu Ülikoolis kevadsemestril toimuv eestikeelne õppeaine, mis kestab seitse nädalat ning on mõeldud erinevate erialade üliõpilastele väljaspool arvutiteaduse instituuti [1]. See kursus on jätkukursus kursusele ”Programmeerimise alused” [2] ja annab üliõpilastele süvendatud alusteadmisi programmide koostamisest, testimisest ja silumisest. Kursus sisaldab seitset uut põhiteemat [3], mille omandamiseks kasutatakse programmeerimiskeelt Python. Kursuse hindamine põhineb koduülesannete ja praktikumi ülesannete, testide, projekti, kontrolltöö ja eksami tulemustel. Kursusel kasutatakse põimõpet, mille puhul kursus toimub mitte ainult auditooriumis, vaid ka veebipõhises keskkonnas, ja pööratud õpet, kus uue teemaga tutvumine toimub väljaspool auditooriumi, nii et üliõpilane uurib materjali oma tempos ning lahendab koduülesandeid. Seejärel tuleb ta auditooriumi, kus kinnistab materjali praktikumi ülesannete abil.

Ülesannetel on kursusel “Programmeerimise alused II” oluline roll õppimises ja nende teksti eest tuleb hoolt kanda, et need oleksid tõhusamad. Murelahendajad mängivad samuti olulist rolli ning koosnevad järjestikustest küsimustest koos vihjete ja koodinäidetega, mis on mõeldud selleks, et aidata üliõpilastel toime tulla erinevate probleemkohtade lahendamisega ülesannetes.

Bakalaureusetöö eesmärk on luua Tartu Ülikooli kursusele “Programmeerimise alused II” uued ülesanded ja neile vastavad murelahendajad. Lõputöö raames koostati 39 ülesannet ning nende jaoks 26 murelahendajat. Esimese ja teise õppenädala ülesannete ja murelahendajate hindamiseks viidi läbi ka üliõpilaste küsitlus.

Lõputöö koosneb viiest peatükist. Esimeses peatükis tutvustatakse kursust “Programmeerimise alused II” ning selle korraldust, õppevormi ning kasutatavat programmeerimiskeelt. Teises peatükis antakse ülevaade ülesannetest, nende õpikeskkonnast ning sellest, kuidas ülesandeid korrektselt struktureerida ja vormistada. Kolmandas peatükis selgitatakse murelahendajate olemust ja eesmärki ning kirjeldatakse nende keskkonda. Neljandas peatükis kirjeldatakse ülesannete ja murelahendajate koostamise protsessi ning ülesannete ja murelahendajate hindamiseks kasutatud tagasiside küsimustikku. Viiendas peatükis analüüsitakse kursuse jaoks loodud ülesandeid ja murelahendajaid. Töö lõpus on lisad, mis sisaldavad ülesandeid, murelahendajaid ja tagasiside küsimustikku.

Bakalaureusetöös on kasutatud tehisintellekti ChatGPT, et eemaldada tekstidest õigekirja- ja grammatikavead ning muuta lausete sõnastust paremaks. Tehisintellekti vastuseid käsitleti kriitiliselt, kuna kõik parandused ei olnud asjakohased.

# 1. Kursus “Programmeerimise alused II”

## 1.1 Kursuse kirjeldus

Õppeinfosüsteemis II (lüh ÕIS II) [1] on kirjas, et kursus “Programmeerimise alused II”, mille ainekood on MTAT.03.256, on Tartu Ülikooli eestikeelne õppeaine, mis annab üliõpilastele süvendatud alusteadmisi programmide koostamisest, testimisest ja silumisest. Kursusel tutvustatakse põhilisi programmeerimiskonstruktsioone, andmetüüpe ja andmestruktuure ning õpetatakse ka programme analüüsima ja algoritme looma lihtsate ülesannete lahendamiseks. Õppeaine on mõeldud erinevate erialade üliõpilastele väljaspool arvutiteaduse instituuti (ingl *Institute of Computer Science*). Õppeaine toimub kevadsemestril ja 2023. aastal registreerus sellele 32 üliõpilast. Kursusest on olemas ka ingliskeelne versioon nimega “Introduction to Programming II”, mida õpetatakse sügissemestril.

Kursus “Programmeerimise alused II” on jätkukursus “Programmeerimise alused” nimelisele kursusele, mis toimub ka kevadsemestril eesti keeles. Kursus “Programmeerimise alused” sisaldab selliseid teemasid nagu andmetüübid, muutujad, sisend kasutajalt, tingimuslaused, tsüklid, järjendid, funktsioonid, andmevahetus ja graafika [2]. Kursus “Programmeerimise alused II” eeldab programmeerimise algteadmisi, mida kursus “Programmeerimise alused” annab [3]. “Programmeerimise alused II” on eeldusaineks paljudele järgmistele õppeainetele nagu “Andmeturve”, “Eesti keele töötlemine Pythonis”, “Keeletehnoloogia”, “Objektorienteeritud programmeerimine”, “Programmeerimine II”, “Programmeerimine koolis (Python)” ja teised.

## 1.2 Programmeerimiskeel Python

Python on objektorienteeritud ja interpreteeritav programmeerimiskeel [4, 5], mille lõi Hollandi programmeerija Guido van Rossum koos teiste autoritega 1990. aastal [6] ning selle esimene versioon ilmus aasta hiljem, 1991. aastal. Viimase 33 aasta jooksul on arendatud mitmeid selle keele versioone. Detsembris 2008. aastal loodi Pythoni versioon 3 [7], mis on leidnud laialdast kasutust peamise programmeerimiskeelena. Programmeerimiskeelt Python ehk Python3 kasutatakse Tartu Ülikoolis mitmetel kursustel ja selle kasutamist õpetab ka kursus “Programmeerimise alused II” [3]. Lõputöö kirjutamise ajal on selle uusim teadaolev versioon 3.12.2, mis ilmus 6. veebruaril 2024. aastal [7].

Python on viimasel ajal muutunud laialt levinud ja populaarseks programmeerimiskeeleks. TIOBE [8] ja PYPL indeksi [9] järgi on Python 2024. aasta aprilli seisuga populaarsuselt esikohal maailmas, edestades teisi keeli nagu C++ ja Java. Lisaks kasutatakse Pythonit erinevates valdkondades nagu andmeanalüüs, tarkvaraarendus, veebirakendused, graafilised liidesed ja süsteemihaldus [5, 10], samuti tekstitöötlus, masinõpe ja närvivõrkude loomine, mis kinnitab selle mitmekülgsust.

Pythoni jaoks on olemas integreeritud programmeerimiskeskkond (ingl *integrated development environment* ehk *IDE*) nimega Thonny, mis on mõeldud algajatele programmeerijatele [11]. Thonny on tasuta ja avatud lähtekoodiga rakendus, mille lõi Eesti programmeerija Aivar Annamaa Tartu Ülikooli arvutiteaduse instituudis koos Raspberry Pi Sihtasutuse (ingl *Raspberry Pi Foundation*) ja Cybernetica ASiga [12, 13]. Programmi saab installida Windowsi, Linuxi ja Maci operatsioonisüsteemidesse [11] ning see on eelinstallitud ka Raspberry Pi operatsioonisüsteemidele [14], muutes selle kergesti kättesaadavaks. Thonnyt kasutatakse kursustel “Programmeerimine” [15], “Programmeerimise alused” [2] ning “Programmeerimise alused II”.

## 1.3 Kursuse korraldus

Kursus “Programmeerimise alused II”, nagu ka sellele eelnev kursus “Programmeerimise alused”, hõlmab Pythoni programmeerimiskeele põhiteemasid. Kursus koosneb seitsmest võtmeteemast, milleks on kahemõõtmeline järjend, kahekordne tsükkel, andmestruktuurid, viitamine ja muteerimine, testimine ja silumine, rekursioon ja objektorienteeritud programmeerimine [3]. Lisaks sisaldab kursus sissejuhatust, mis valmistab ette põhiteemade käsitlemiseks.

Kursus toimub Moodle'i keskkonnas [16] ning kestab seitse nädalat. Iga teema kohta on videoloengud, kuid ei toimu kontaktloenguid. Videoloengud on 5-25 minuti pikkused videomaterjalid, milles õppejõud räägib kursuse põhiteemast ja demonstreerib koodinäiteid. Igal üliõpilasel on võimalus vaadata videot endale sobival ajal. Samuti on iga teema kohta ka foorum, kus üliõpilased saavad esitada küsimusi või arutada tekkinud probleeme.

Kursuse „Programmeerimise alused II“ hindamine on eristav, mis tähendab, et üliõpilaste õpitulemuste saavutamist eristatakse skaala alusel (hinded A, B, C, D, E, F). Tabelis 1 on esitatud punktide jaotumine kursusel [3].

**Tabel 1.** Kursuse “Programmeerimise alused II” hindamisskeem.

| | **Minimaalne punktide arv** | **Maksimaalne punktide arv** |
|---|---|---|
| Programmeerimisülesannete kodutööd + praktikumid | 8 | 15 |
| Projekti ülesande püstitus | | 3 |
| Projekti ülesande lahendus | Projekti püstitus + lahendus 10 | 17 (sellest 3 aruanne ja 3 esitlusvideo) |
| Testid | 7 | 17 |
| Arvutikontrolltöö | 13 | 26 |
| Eksamitöö | 13 | 26 |

Õppeaines “Programmeerimise alused II” on kodutööd, praktikumid, testid, projekt, kontrolltöö ja eksamitöö. Kokku on kursusel võimalik koguda maksimaalselt 104 punkti. Minimaalse positiivse hinde saamiseks peab üliõpilane koguma vähemalt 51 punkti ja täitma kõik lävendid.

Kodutööde ja praktikumide eest (tabel 1) on võimalik saada kokku maksimaalselt 15 punkti ning lävend on 8 punkti. Kokku on viis kodutööd, kus esimesed kolm sisaldavad igaüks kolme erinevat ülesannet, samas kui kahes viimases on kummaski kaks ülesannet. Iga kodutöö eest antakse 2 punkti ja üliõpilane peab need sooritama enne praktikumi toimumist. Praktikumid toimuvad kohapeal Tartu Ülikooli Delta keskuse majas [18] või eemalt Zoom’i kaudu [19]. Üliõpilane saab ühe punkti iga praktikumi eest, kui on sooritanud vähemalt ühe ülesande. Kodutööde eest on võimalik saada kokku 10 punkti ja praktikumide eest 5 punkti, kokku seega 15 punkti.

Kursusel tuleb teha ka projekt, mis on iga üliõpilase individuaalne töö. Enne kolmandat praktikumi peab üliõpilane esitama projekti ülesande püstituse tekstifailina, ja enne kuuendat praktikumi projekti programmi, aruande ja esitlusvideo. Projekti ülesande püstituse eest (tabel 1) on võimalik saada maksimaalselt 3 punkti, projekti ülesande lahendamise eest 17 punkti ning lävend on 10 punkti.

Kursusel on ka testid (tabel 1), mille eest võib saada maksimaalselt 17 punkti ja lävend on 7 punkti. Testid jagunevad Moodle'i keskkonnas olevateks testideks, mida on kokku 8, ja testideks, mis on videoloengu küsimused. Moodle’i testid 1-4, 6 ja 7 annavad 1 punkti ja testid 5 ja 8 annavad 2 punkti, ning kokku 10 punkti. Videoloenguid on kokku 13, kuid neist

ainult 7 on ülesandega ja iga videoloengu testi eest saab 1 punkti. Moodle'i testide eest antakse 10 punkti ja videoloengu küsimuste eest 7 punkti, mis annab kokku 15 punkti.

Lisaks on kursusel (tabel 1) arvutikontrolltöö ja eksam. Mõlemast tööst võib saada maksimaalselt 26 punkti, lävendiks on 13 punkti. Kontrolltöö kestab 100 minutit ning koosneb kahest ülesandest, millest kumbki annab 13 punkti. Ülesanded lahendatakse arvutis, kasutades Thonny programmeerimiskeskkonda ja vajadusel materjale. Eksam kestab 60 minutit ning toimub Moodle'i testi vormis, mis koosneb 21 küsimusest. Küsimused lahendatakse arvutiklassi arvutis, ilma abimaterjalideta, välja arvatud üks A4-paberileht. Mõlema töö puhul on igale rühmale auditooriumis osalemine kohustuslik, samuti on keelatud suhtlemine ja tehisintellekti kasutamine.

## 1.4 Kursuse õpetamine

Vajalik on käsitleda kursuse õpetamisega seotud aspekte, kuna need on otseselt seotud ülesannetega. Moodle'i keskkonna põhjal [16] tutvuvad ülipilased kursusel ”Programmeerimine alused II” materjalidega enne praktikumi toimumist. Kui samal teemal on kodutööd ja praktikumi ülesanded, siis peab üliõpilane esmalt sooritama koduülesanded ja alles seejärel tegelema praktikumi ülesannetega tunnis. Selline lähenemine õpetamisele erineb traditsiooniliselt koolis kasutatavast lähenemisest. Tavaliselt tutvustatakse uut teemat klassis, kus kohe lahendatakse ka esimesed ülesanded, ja kodutöid tehakse hiljem kodus. Kuigi kursuse korraldus tundub esmapilgul ebatavaline, on see tegelikult tihedalt seotud selliste õppevormidega nagu põimõpe ja pööratud õpe.

Õppevorm (ingl *form of study*) on õppimise korraldamise viis konkreetse õppekava, kursuse või õppeaine raames [17, 18]. Hariduskeskkonnas on palju erinevaid õppevorme, igaühel oma meetod või lähenemine õppeprotsessile. Sobiva õppevormi valik sõltub nii kursuse eesmärkidest kui ka korraldusest. Tartu Ülikoolis jagunevad õppeained kolme peamise õppevormi vahel: lähiõpe (ingl *face to face learning*), veebiõpe (ingl *online learning*) ja põimõpe. ÕIS II keskkonnas [1] on iga kursuse lehel jaotisest “Õpikeskkond” võimalik leida, millist kolmest õppevormist kursus kasutab.

### 1.4.1 Põimõpe

Põimõpe (ingl *blended learning*) on õppevorm, kus osa õppest toimub lähiõppena ühises auditooriumis ja osa kaug- või veebiõppena [17, 18]. Garrisoni ja Kanuka järgi [19] võib põimõpe, mis ühendab lähi- ja veebiõpet, avaldada positiivset mõju ja parandada

kõrghariduse kvaliteeti, kui see on läbimõeldult integreeritud. López-Pérez jt uuringu tulemustest selgus [20], et põimõppe kasutamine kõrghariduse kursustel avaldas positiivset mõju üliõpilaste tulemuslikkusele, vähendades märkimisväärselt väljalangevuse määra ja parandades nende eksamihindeid. Tänapäeval on põimõpe kõrghariduses väga kasulik ja seda kasutatakse paljudes valdkondades, sealhulgas programmeerimises.

ÕIS II keskkonna põhjal [1] toimub kursus "Programmeerimise alused II" põimõppena, samuti nagu teised programmeerimiskursused, näiteks "Programmeerimise alused", "Programmeerimine" ja "Objektorienteeritud programmeerimine". Neid kursusi ühendab see, et praktikumid, kontrolltööd ja eksamid viiakse läbi auditooriumis ning loengud, õppematerjalid ja testid on kättesaadavad veebis.

Stakeri ja Horni järgi [21] on olemas neli põimõppe mudelit ning pööratud õpe on üks variantidest, kuidas rakendada rotatsioonimudelit (ingl *rotation model*) põimõppes. Rotatsiooni mudel on lähenemine õppimisele ning üks neljast põimõppe mudelist, mille puhul üliõpilased vahelduvad kindla ajakava alusel või õppejõu äranägemisel erinevate õppevormide vahel, millest üks on veebiõpe [21]. Sellest järeldub, et pööratud õpe on põimõppe üks vorm.

### 1.4.2 Pööratud õpe

Pööratud õpe ehk klassiruum (ingl *flipped classroom*, *flipped learning*, *inverted classroom*), tuntud ka kui ümberpööratud õpe ehk klassiruum, on õppevorm, mille puhul üliõpilased omandavad teadmisi kõigepealt väljaspool auditooriumi omal ajal, peamiselt loenguid vaadates, ning seejärel harjutavad ja kinnistavad materjali auditooriumis [22]. Pööratud õpe on traditsioonilise õppe vastand, mida esmakordselt tutvustasid 2000. aastal Baker [23] ning Lage jt [24]. Nende kohaselt peaksid sündmused, mis varem toimusid klassiruumis, nüüd toimuma väljaspool, ja vastupidi. Bishopi ja Verlegeri järgi [25] ei ole pööratud õpe lihtsalt sündmuste ümberpaigutamine, vaid see on terve õppekava laiendus, mis hõlmab kahte peamist komponenti: esiteks, individuaalne arvutipõhine õpe väljaspool klassiruumi, mis sisaldab videoloenguid, teste või ülesandeid, ning teiseks, interaktiivsed rühma õppetegevused klassiruumis.

Al-Samarraie jt järgi [26] on pööratud õpe leidnud rakendust erinevates kõrgharidusvaldkondades, sealhulgas tehnika ja tehnoloogia, matemaatika, meditsiin ja tervishoid, sotsiaal- ja humanitaarteadused, samuti loodusteadused nagu keemia ja bioloogia.

Pööratud õpe on muutunud populaarseks ka programmeerimise õpetamisel. Rosiene CP ja Rosiene JA poolt läbi viidud uuringus [27] tuvastati pööratud õppe tugevad ja nõrgad küljed sissejuhatavas programmeerimiskursuses. Üheks positiivseks aspektiks on võimalus, et üliõpilased saavad õppida individuaalses tempos – nad saavad videoloenguid edasi- ja tagasi kerida, kiirendada, vahele jätta ja pausile panna. Samuti säästab õppejõud aega auditooriumis, kuna loengud ei ole kohapeal, nii et seal saab pühenduda ainult aktiivsele õppimisele. Negatiivsete aspektide hulka kuuluvad üliõpilaste võimalik motivatsioonipuudus materjali õppimisel; testid, mida üliõpilased sooritavad ilma järelvalveta, ning küsimused, mida üliõpilased saavad esitada ainult klassiruumis, kuid vaatamata neile peetakse pööratud õpet heaks õppevormiks [27].

Pööratud õpe on leidnud laialdaselt rakendust ka Tartu Ülikoolis. Lepp ja Tõnisson viisid 2015. aastal läbi uuringu [28], mis näitas, et pööratud õppe kasutuselevõtt kasutusele kursusel "Objektorienteeritud programmeerimine" [29] koos paaristööga praktikumide ajal. Üliõpilaste küsitluse tulemused olid positiivsed ja enamik oli kursusega rahul [28]. Hiljem hakati pööratud õpet kasutama teistel programmeerimiskursustel nagu "Programmeerimise alused" [2], "Programmeerimise alused II" [3] ja "Programmeerimine" [15]. Kursusel "Programmeerimise alused II" [3] alustatakse materjaliga tutvumist ja testide sooritamist ning alates teisest õppenädalast lisanduvad veebis lahendatavad koduülesanded. Neile järgnevad ülikoolis toimuvad kohtumised, kus üliõpilased praktikumide käigus kordavad ja kinnistavad õpitud teemasid.

# 2. Ülesanded

## 2.1 Ülesanded ja nende väärtus

Kursustel on ülesandeid erinevatel teemadel, mida õpilastel ja üliõpilastel palutakse lahendada. Kursusel “Programmeerimise alused II” mängivad ülesanded keskset rolli, kuna need on suunatud programmeerimisoskuste arendamisele. Kursusel peavad üliõpilased lahendama kodu- ja praktikumi ülesandeid seitsme teema kohta, mille eest nad saavad punkte, mis lähevad arvesse kursuse lõpphinde kujunemisel [3]. Ülesannetes nõutakse, et üliõpilased koostaksid erinevaid programme, kirjutades koodi, ja esitaksid need seejärel kontrollimiseks.

Ülesanded omavad olulist väärtust õppeprotsessis. Weinmani järgi [30] võimaldavad kodu- ja praktikumi ülesanded kursuste ajal üliõpilastel uurida loengus käsitletud kontseptsioone ja nende abil praktiseerida määratud probleemide lahendamist. VanDeGrift järgi [31] mõistavad üliõpilased programmeerimist kõige paremini, kui nad loovad programme väljaspool tundi. Edgcombi jt uuringute järgi [32] on üliõpilastel sageli raske loenguteks valmistuda, kuid koduülesanded pakuvad neile selleks vajalikku tuge. Epsteini ja Voorhise järgi [33] annavad õpetajad koduülesandeid mitmetel eesmärkidel. Näiteks valmistavad koduülesanded üliõpilasi ette järgmisteks tundideks ja tulevasteks testideks, kontrollivad ja säilitavad oskusi, arendavad vastutustunnet ja püsivust ning suurendavad õppimisse kaasamist [33]. Lisaks aitavad koduülesanded parandada materjali mõistmist ja meeldejätmist [34, 35], tugevdavad iseseisva töö oskusi ja soodustavad kriitilise mõtlemise arengut [34]. Eespool toodud uuringute põhjal on koduülesannetel palju positiivseid aspekte, sealhulgas ettevalmistus järgmisteks õppetundideks ja materjali mõistmise tugevdamine. Vaatamata sellele, et enamik allikad keskendusid ainult koduülesannetele, võib sarnaseid eeliseid kohaldada ka praktikumi ülesannete suhtes, mis rõhutab nende tähtsust õppeprotsessis ja näitab, kuidas praktikumi ülesanded toetavad süvendatud õppimist ja praktiliste oskuste arendamist.

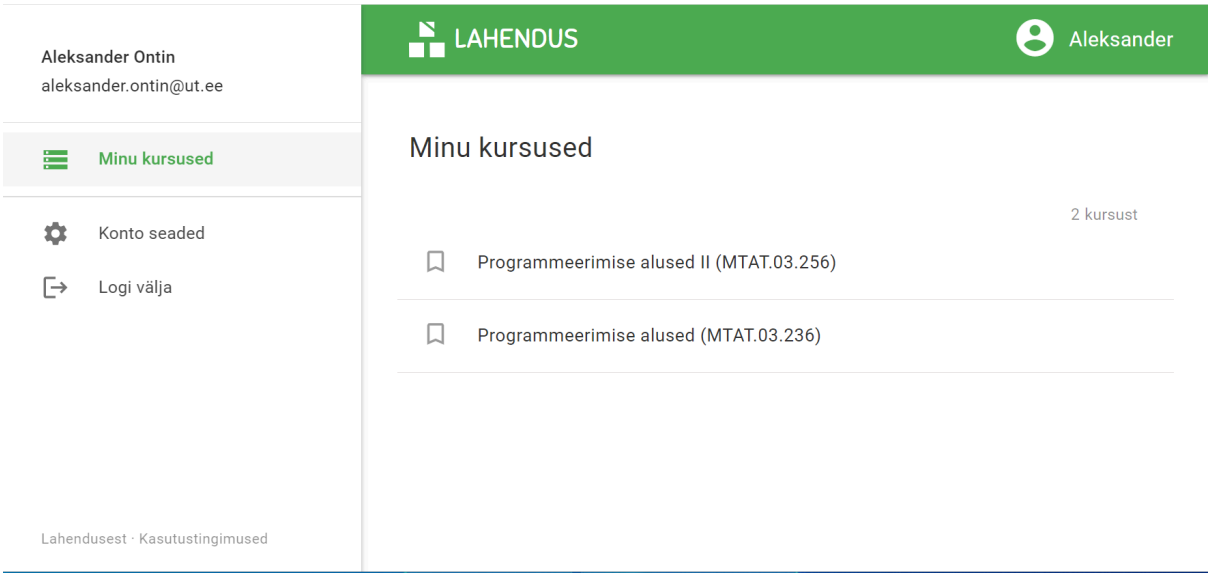
Ülesannete puhul on oluline mitte ainult nende koostamine, vaid ka üliõpilaste kaasamine nende lahendamisse. Üks võimalus üliõpilasi ülesandeid lahendama suunata on premeerida neid selle eest punktidega. VanDeGrifti uuringust selgus [31], et enamik üliõpilasi lahendab koduseid ülesandeid, kuna nende eest saab hindeid, mis mõjutavad kursuse lõpptulemust. Samuti mõjutab MOOCide (ingl *Massive Open Online Course*) õppematerjal oluliselt üliõpilaste motivatsiooni ülesannete lõpetamisel. Huang, B. ja Hew, K. F. T. uuringu

tulemustest selgus [36], et enamik üliõpilasi olid õppematerjalidega rahul, leides neid kasulikeks, mis omakorda suurendas nende motivatsiooni. Tasub märkida, et lisaks punktide saamise süsteemile ja õppematerjalide kvaliteedile võivad üliõpilaste motivatsiooni suurendada ka interaktiivsed õpimeetodid, grupitööd ja reaalsete probleemide lahendamine.

## 2.2 Ülesannete keskkond

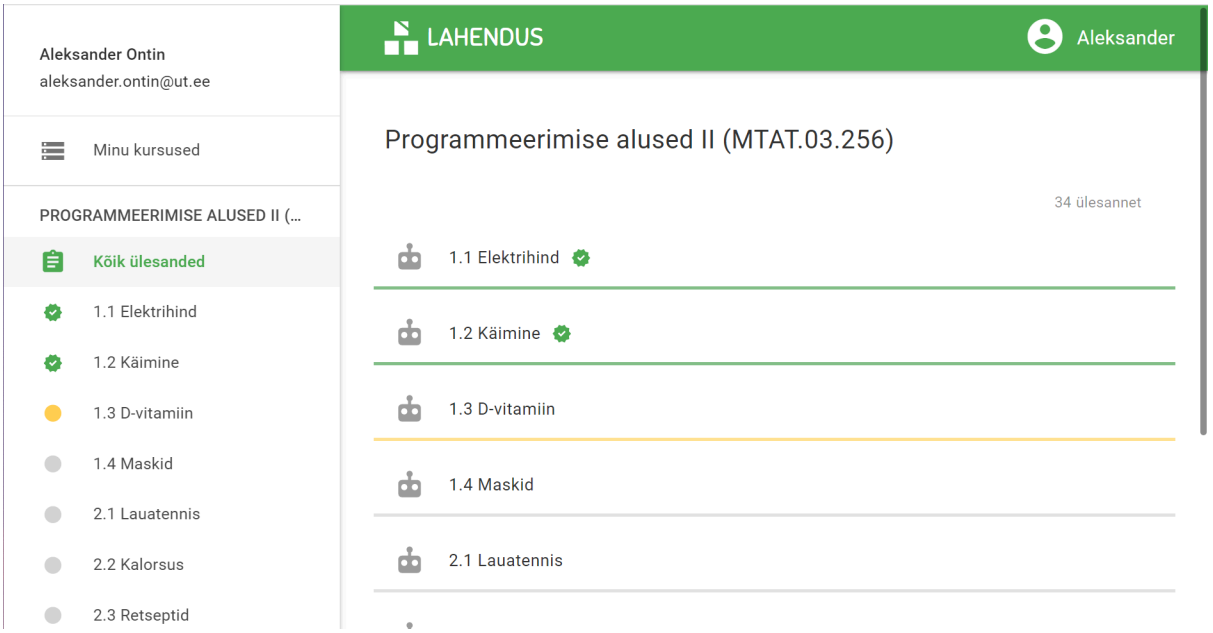
Kursusel "Programmeerimise alused II" kasutatavad ülesanded paiknevad õpikeskkonnas Lahendus, mis asub aadressil https://lahendus.ut.ee [37]. Üliõpilased pääsevad nendele ülesannetele vahetult ligi otse Moodle'i õppesüsteemi kaudu, mis on kursusega integreeritud.

Lahendus on avatud lähtekoodiga veebirakendus ning programmeerimise õppimise ja õpetamise keskkond eesti ja inglise keeles, mille omanik ja haldaja on Tartu Ülikooli arvutiteaduse instituut [37, 38]. Käesoleva lõputöö kirjutamise ajal on Lahenduse keskkonnas üle kuue tuhande registreeritud kasutaja. Lahenduse keskkonna kasutamiseks tuleb luua konto, saades seeläbi Lahenduse kasutajaks. Registreeritud õpetajatele ja õppejõududele pakutakse vahendeid programmeerimiskursuste loomiseks ja haldamiseks, sealhulgas üliõpilastele ülesannete koostamiseks. Üliõpilased, kes liituvad Lahenduse keskkonnas vastava kursusega, saavad vaadata ja lahendada pakutud ülesandeid, mille eeliseks on automaatkontroll, tänu millele saab kontrollida, kas ülesanne on hästi lahendatud või mitte. Lisaks on õpikeskkonnal ametlik Discordi server nimega "easy" [39], kus kasutajad saavad teatada probleemidest või esitada küsimusi veebirakenduse kohta.



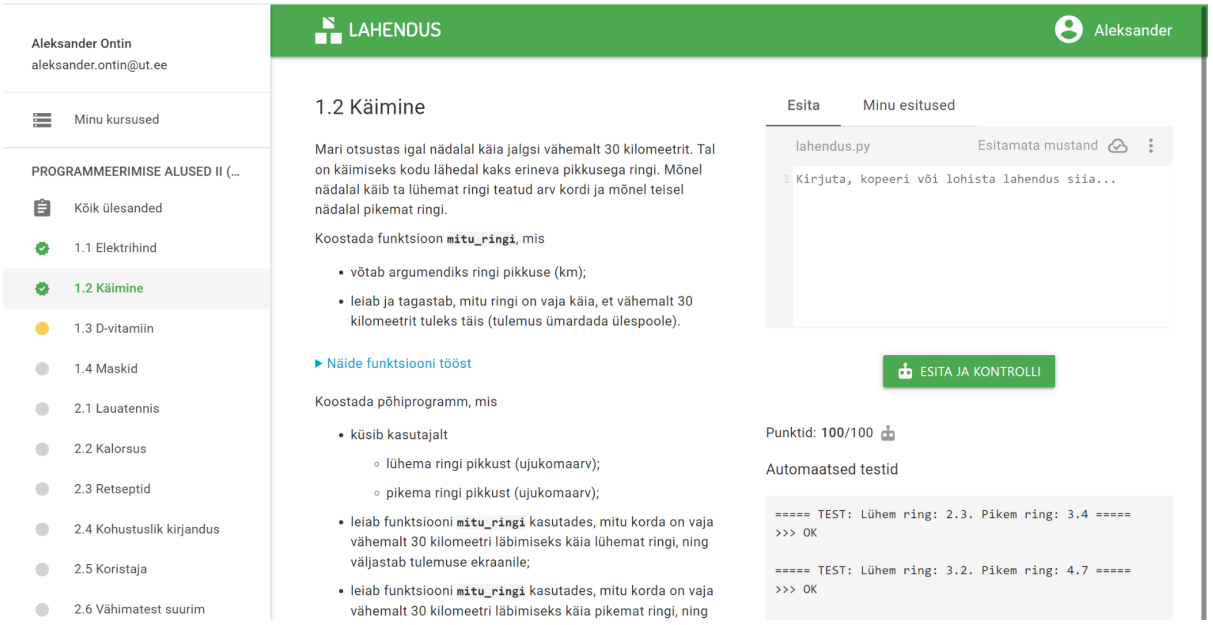

Joonis 1. Avalehe "Minu kursused" vaade.

Pärast registreerimist leiab kasutaja end avalehelt “Minu kursused” (joonis 1). Avalehe keskel kuvatakse nimekiri programmeerimiskursustest, millele kasutaja on registreerunud. Vajutades kursuse nimele, saab soovitud kursuse juurde liikuda (joonis 2). Vasakul on navigatsioonikomponent, mida nimetatakse Külgmenüü (ingl *Slide-out menu*). Tahvelarvuti või mobiiltelefoni kasutamisel [40] on navigatsioonikomponent peidetud ja selle kuvamiseks tuleb vajutada Hamburgeri menüü nupule (ingl *Hamburgeri menu*). Navigatsioonikomponent sisaldab linke, millega saab liikuda veebirakenduse erinevatele lehtedele, nt kasutajakonto seaded, väljalogimine, avaleht kursustega, veebirakenduse infoleht ja erinevaid tingimusi sisaldav dokument.



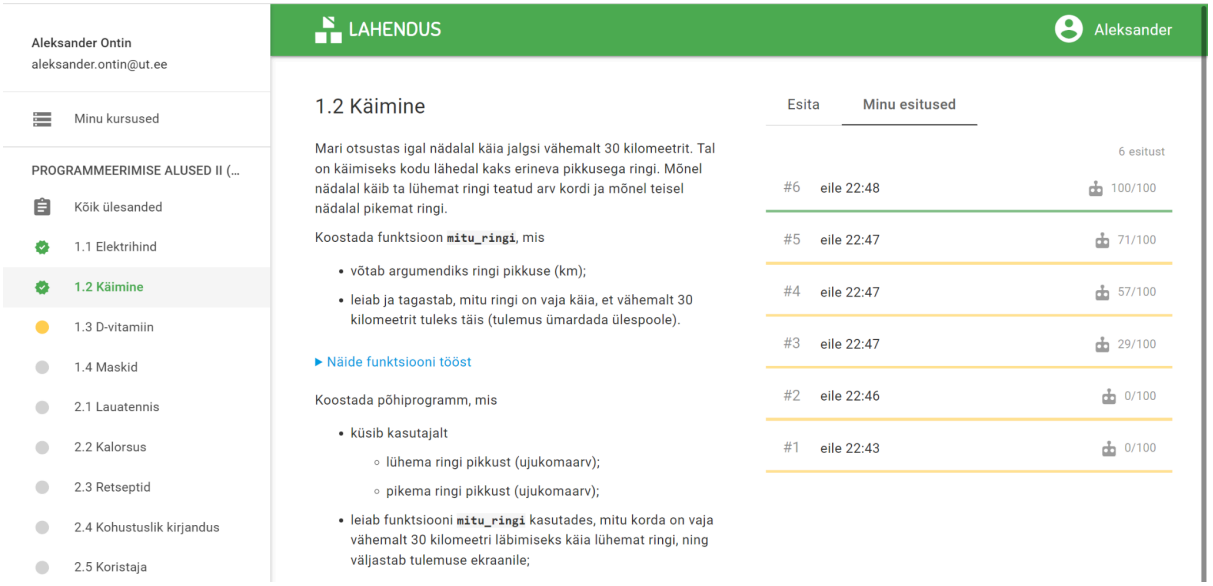

Joonis 2. Kursuse “Programmeerimise alused II (MTAT.03.256)” vaade.

Kursuse “Programmeerimise alused II (MTAT.03.256)” lingile vajutades jõuab kasutaja kursuse lehele (joonis 2). Kursuse lehel leidub nummerdatud ülesannete nimekiri. Ülesande nimele vajutades liigub kasutaja konkreetse ülesande lehele. Navigeerimiskomponendis on uus nupp “Kõik ülesanded”, mis viib ülesannete lehele. Allpool asuvad üksikute ülesannete nupud, mis samuti suunavad konkreetsetele ülesannete lehtedele. Listis ja navigatsioonikomponendis on iga ülesande nime kõrval märge, mis näitab selle lõpetamise tulemust. Roheline värv tähendab, et ülesanne on lõpetatud 100 punktiga. Kollane värv osutab, et ülesanne on osaliselt lõpetatud või lahendus esitatud, kuid mitte kõik automaatsed testid ei ole õnnestunud. Hall värv näitab, et kasutaja ei ole ülesande lahendust veel esitanud.



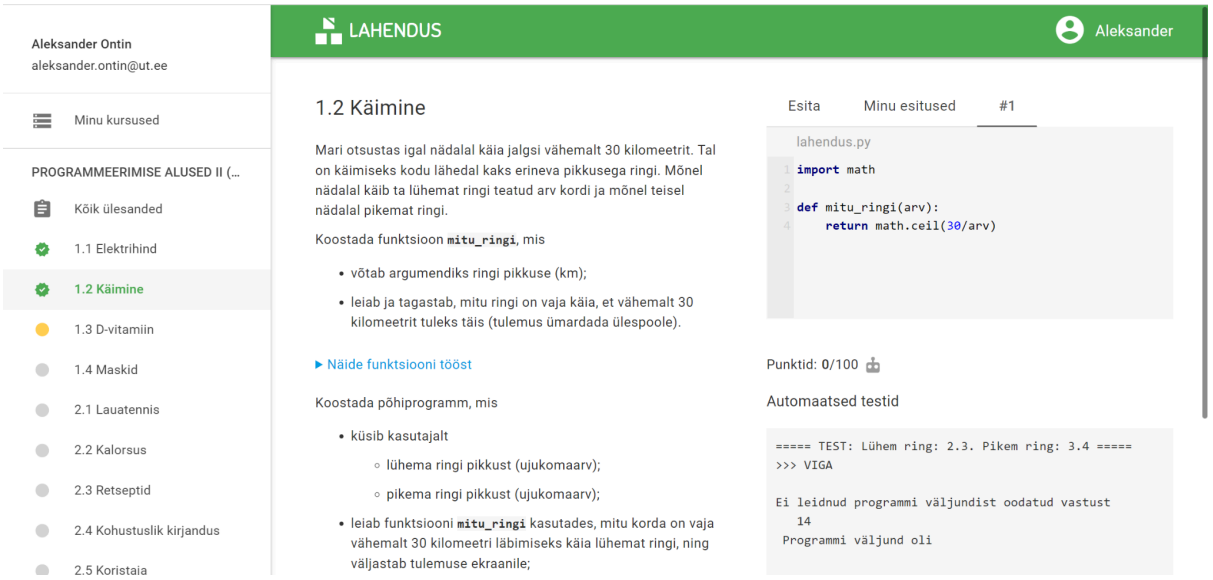

Joonis 3. Kursuse “Programmeerimise alused II (MTAT.03.256)” teise ülesande vaade.

Vajutades konkreetse ülesande lingile, jõuab kasutaja ülesande lehele (joonis 3), kus kuvatakse valitud ülesande nimi ja allpool selle täielik kirjeldus. Paremal pool asub navigatsioonikomponent nimega Horisontaalsed vahekaardid (ingl *Horizontal tabs*). Selles on kaks nuppu: “Esita” ja “Minu esitused”, mis võimaldavad vahetada teabeosi. Vahekaardil “Esita” peab kasutaja kirjutama või kleepima koodi ja seejärel vajutama nupule “ESITA JA KONTROLLI”. Pärast nupuvajutust kuvatakse allpool tulemused vahemikus 0 kuni 100 punkti, mis sisaldavad automaatseid teste erinevate sisendväärtustega. Testide tulemused näitavad “OK”, kui testid on edukad, või “VIGA”, kui testid ebaõnnestuvad.




Joonis 4. Kursuse “Programmeerimise alused II (MTAT.03.256)” teise ülesande vaade.

Vahekaart “Minu esitused” (joonis 4) sisaldab kõiki kasutaja poolt konkreetse ülesande kohta esitatud lahendusi. Esitatud lahendused on järjestatud esitamise aja järgi kahanevas järjekorras. See tähendab, et kõige esimene esitatud lahendus asub nimekirja allosas ja kõige hiljutisem lahendus ülaosas. Igal esitatud lahendusel on oma number, lahenduse esitamise aeg ja punktide arv. Kui lahendus on korrektne, on selle all olev joon roheline; kui lahendus on osaliselt korrektne või vale, on joon kollane.




Joonis 5. Kursuse “Programmeerimise alused II (MTAT.03.256)” teise ülesande vaade.

Kui kasutaja vajutab lahenduse esitamise ajale, siis ilmub üks uus vahekaart koos numbriga (joonis 5). Sellele vahekaardile vajutades saab näha koodi, punktide arvu ja kõiki automaatseid teste, mida kasutati kontrollimiseks.

## 2.3 Ülesannete teksti koostamine

Ülesannete koostamisel on oluline mõelda teksti kirjutamisele ja kasutamisele, sest see on see, mida üliõpilane näeb enne ülesannete lahendamise juurde liikumist. Nagu e-kursuste puhul [41], on ka ülesannete tekstid peamised vahendid informatsiooni edastamiseks üliõpilastele. Selles alapeatükis arutletakse kursuse “Programmeerimise alused II” kasutatava ülesannete struktuuri ja teksti kujundamise üle.

Ülesannete koostamisel mängib struktuur olulist rolli. On tähtis, et ülesanne oleks sidus ja hästi struktureeritud. Teave ülesande struktuuri kohta põhineb eelmise aasta ülesannetel Lahenduse keskkonnas [37]. Struktuurilisest vaatepunktist võib ülesande teksti jagada

mitmeks osaks, mis peavad järgima loogilist järjekorda ja millest mõned on kohustuslikud. Enamik selle kursuse ülesannete tekste on struktureeritud järgnevalt:

- pealkiri
- taustainfo
- nõuded funktsiooni kohta (kui ülesandes on funktsioon nõutud)
- näited funktsiooni tööst (kui ülesandes on funktsioon nõutud)
- nõuded programmi kohta
- näited programmi tööst
- vihjed
- abi

Programmeerimisülesanded peaksid üldjuhul algama pealkirjaga. Hea pealkiri kajastab ülesande põhiolemust, aidates üliõpilastel aru saada, mida see käsitleb. Lisaks on igale ülesandele enne pealkirja lisatud järjekorranumber (joonis 2), mis aitab ülesandeid kergemini leida.

Ülesande koostamisel võib olla vajalik kasutada taustainfot (ingl *background*), et aidata üliõpilastel mõista ülesande konteksti. Taustainfo sisaldab põhiteavet, mida üliõpilane peab ülesande lahendamiseks kasutama. See osa hõlmab tavaliselt ühe lõiku, kuid võib ulatuda ka mitme lõiguni. See osa on kohutuslik, kui ülesandes on detaile, mis võivad olla üliõpilasele ebaselged. Samuti on see vajalik, kui on oluline rõhutada ülesande praktilist tähtsust ja asjakohasust. Teksti saab rikastada linkidega, mis suunavad erinevatele veebilehtedele ja sisaldavad asjakohast teavet ülesande jaoks.

Nõuded funktsiooni kohta kirjeldavad konkreetseid tegevusi, mida funktsioon peab täitma. Funktsiooni nõuded peavad selgelt määratlema, millised argumendid on vajalikud ning mida funktsioon peab tegema ja/või tagastama. Lisaks on vajalik lisada vahetult pärast nõudeid näited funktsiooni tööst. Näited on olulised funktsiooni vaatamiseks ja testimiseks, ning nende arv sõltub ülesande nõuetest. Selgelt sõnastatud nõuded ja näited funktsiooni kohta aitavad üliõpilastel paremini mõista, mida neilt oodatakse.

Nõuetel programmi kohta on sama roll kui nõuetel funktsiooni kohta. Programm võib küsida kasutajalt sisendit, lugeda andmeid failist, kasutada funktsiooni või väljastada ekraanile teate. Kui ülesanne nõuab, et üliõpilased koostaksid lisaks programmile ka funktsiooni, siis peaksid programmi nõuded järgnema funktsiooni nõuetele ja näidetele. Pärast nõudeid programmi kohta on lisatud ka näiteid selle kohta, kuidas programm töötab. Sarnaselt funktsiooni

näidetega, aitavad programmi näited üliõpilastel kiiremini mõista, mida programm teeb ja kuidas see töötab. Kui programm nõuab failist lugemist, peaks näidisprogramm näitama ka seda, milline peaks olema failis olev tekst.

Ülesannete teksti koostamisel võib mõnikord osutuda vajalikuks anda vihjeid. Vihjetes võib mainida funktsioone, mis on ülesande lahendamiseks kasulikud, sealhulgas neid, mida ei ole materjalides mainitud, kuid mis võivad olla üliõpilastele abiks. Vihjeid tuleks lisada pärast näiteid programmi või funktsiooni töötamise kohta.

Ülesande lõpus on oluline lisada mitmesuguseid abivahendeid, mis aitavad üliõpilastel ülesandeid lahendada. Enamasti on need murelahendajad, mida käsitletakse peatükis "Murelahendajad". Abivahendid võivad sisaldada ka abivideoid, kus rääkija selgitab konkreetse ülesande olemust ja seda, kuidas seda saab lahendada.

Ülesannete koostamisel mängib vormistamine väikest, kuid olulist rolli. Hästi vormistatud ülesannet on lihtsam lugeda ja sellest aru saada. Teave ülesande vormistamise kohta põhineb kursusel "Kursuse loomise ABC" [41], kus käsitletakse teksti kasutamist. Kuigi see informatsioon on otseselt seotud kursuste loomisega, on paljud põhimõtted kohaldatavad ka ülesannete koostamisel.

Ülesande teksti vormistamine koosneb järgmistest osadest:

- teksti lihtsustamine
- täpploendite kasutamine
- teksti rõhutamine
- tekstide fondid

Tähelepanu tuleb pöörata ülesannete teksti lihtsustamisele, sest keeruline tekst võib arusaadavust negatiivselt mõjutada. Chandrasekari ja Srinivase järgi [42] põhjustavad pikad ja keerulised laused mitmeid probleeme, nagu ebatäpne analüüs või segadus, mida nende lihtsustamine võib kõrvaldada või minimeerida. Ülesande tekst peaks olema üliõpilastele kergesti loetav; selleks tuleks laused hoida lühikesed ja selged ning kasutada asjakohaseid termineid. Teksti lihtsustamisel on oluline säilitada ülesande põhiidee ja mitte eemaldada olulist teavet, vastasel juhul võib see muuta ülesande mõistmise keerulisemaks. Teksti lihtsustamine on oluline ülesanne, millesse tuleks suhtuda ettevaatlikult.

Ülesannete teksti koostamisel on kasulik jagada tekst väiksemateks osadeks, kasutades täpploendit (ingl *bulleted list*), mis aitab üliõpilastel tekstist paremini mõista. Näiteks võib

tuua kaks lõiku: esimene ilma täpploendita (näide 1) ja teine täpploendiga (näide 2), mõlemad näited on koostatud ühe eelmise aasta ülesande põhjal [37].

**Näide 1.** Lõik ilma täpploendita

Koostada funktsioon **mitu_ringi**, mis võtab argumendiks ringi pikkuse (km) ning leiab ja tagastab, mitu ringi on vaja käia, et vähemalt 30 kilomeetrit tuleks täis (tulemus ümardada ülespoole).

**Näide 2.** Lõik täpploendiga

Koostada funktsioon **mitu_ringi**, mis

- võtab argumendiks ringi pikkuse (km);
- leiab ja tagastab, mitu ringi on vaja käia, et vähemalt 30 kilomeetrit tuleks täis (tulemus ümardada ülespoole).

Ülesannete jaoks teksti koostamisel on mõistlik kirjutada teatud võtmesõnad paksus kirjas, et neid rõhutada. Rõhutatud sõnad lõigus äratavad üliõpilaste tähelepanu, aidates neil lõiku tähelepanelikumalt lugeda ja mõista selle tähendust. Programmeerimisülesannetes on funktsioonide ja tekstifailide nimed enamasti rasvases kirjas või muul viisil rõhutatud.

Lisaks on ülesannete tekstide koostamisel oluline kasutada turvalisi veebifonte (ingl *web safe fonts*) [43], mis mõjutavad üliõpilaste teksti mõistmist, kuna neid on lihtne lugeda. Keskkonnas Lahendus [37] kasutatakse ülesannete tekstide jaoks fonti Roboto [44], mis on Google'i poolt kasutatud turvaline veebifont. Erinevate koodiridade jaoks kasutatakse fonti Courier New.

# 3. Murelahendajad

## 3.1 Murelahendajad ja nende väärtus

Tööde teostamisel ja korraldamisel võivad sageli esineda probleemid, mis vajavad lahendamist. Tõrkeotsing ehk vigade leidmine (ingl *troubleshooting*) on just üks levinumaid ja laialdaselt kasutatavaid probleemide lahendamise viise, mille puhul kõigepealt tuvastatakse vea põhjus ning seejärel parandatakse see. Jonasseni ja Hungi järgi [45] on tõrkeotsing kõige sagedamini seotud elektrooniliste, mehaaniliste ja füüsiliste süsteemidega, kuid leidub ka teistes valdkondades, nagu klienditeeninduses ning personalijuhtimises, kuid ka meditsiinis ja psühholoogias. Tõrkeotsija ehk veaotsija (ingl *troubleshooter*) on tavaliselt mingi vahend, mis tegeleb probleemide leidmise ja parandamisega teatud valdkonnas. Selles lõputöös keskendutakse sammsammulisele tõrkeotsijale [46] ehk murelahendajale, mida kasutatakse "Programmeerimise alused II" kursusel.

Murelahendajatel on õppimiseks oluline roll. Lepp jt järgi [47] on murelahendajate koostamise peamine eesmärk aidata üliõpilastel lahendada probleeme, mis tekkivad ülesannete lahendamise käigus. Nad pakuvad vihjeid ja koodinäiteid, mis aitavad jõuda õigete lahendusteni ja parandavad arusaamist koodi kirjutamisest. Koostatud murelahendajaid kasutatakse abivahendina, et aidata üliõpilastel lahendada kodu- ja praktikumi ülesandeid. See abivahend koosneb esialgu küsimustest, mis järgivad programmi koostamise järjekorda ning aitavad selgitada programmi probleemseid kohti. Probleemi korral on võimalik saada küsimusest abi selgituse näol ning kui probleem on lahendatud, on võimalik edasi liikuda järgmise küsimuse juurde. Murelahendaja peaks aitama ülesande täielikult lahendada. Kui aga mõni probleem jääb lahendamata, sisaldab murelahendaja lõpus vihjet, mis soovitab üliõpilasel pöörduda kursuse juhendaja poole.

## 3.2 Murelahendajate kasutamine

Murelahendajad võeti Tartu Ülikoolis esmakordselt kasutusele 2016. aasta kevadel, kasutades neid populaarses MOOCis "Programmeerimisest maalähedaselt" [48], milles osaleb suur hulk üliõpilasi [1, 37]. Lepp jt uuringus [47] leiti, et murelahendajate kasutuselevõtuga vähenes kursuse juhendajale saadetud kirjade arv 29% võrra. Küsimustik näitas, et 80,18% osalejatest kasutas murelahendajat ja neist 40,8% pidas seda väga kasulikuks, kuid 3,5% leidsid, et see ei olnud neile abiks. Ülesannete jaoks murelahendajate loomine osutus edukaks, kuna need olid üliõpilastele väga kasulikud [36], mis omakorda suurendas nende populaarsust ja kasutust.

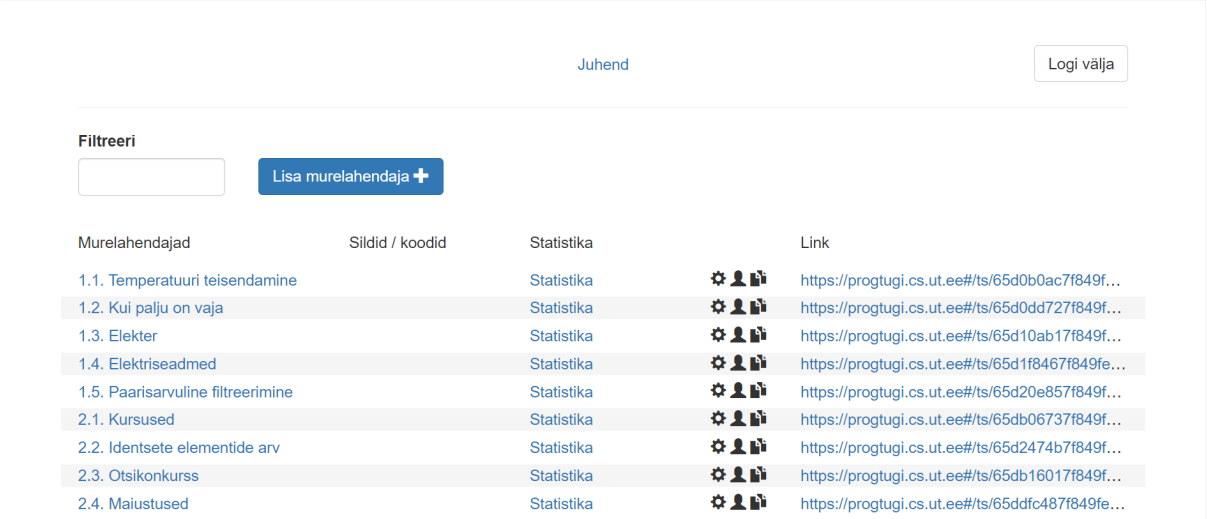
Pärast positiivset mõju esimesele kursusele loodi murelahendajad kiiresti ka kursustele “Programmeerimise alused” ja “Programmeerimise alused II”. Samuti on Tartu Ülikoolis hakanud murelahendajaid koostama üliõpilased erinevate kursuste jaoks oma bakalaureusetööde raames. Näiteks 2020. aastal koostas Mihkel Pent murelahendajad kursuse “Tehnoloogia tarbijast loojaks” jaoks [49], samal aastal Anette Klaanberg kursuse “Objektorienteeritud programmeerimine” jaoks [50]. Samuti 2021. aastal koostas Joosep Kaimre murelahendajad kursuse “Programmeerimine” jaoks [51], 2022. aastal Karolin Kivilaan kursuse “Sissejuhatus andmebaasidesse” jaoks [52] ja 2023. aastal Karl Eric Ott kursuse “Andmebaasid” jaoks [53]. Lisaks koostasid üliõpilased oma bakalaureusetöö raames ka ülesandeid koos murelahendajatega ingliskeelsetele programmeerimiskursustele. Aastal 2020 koostas Taniel Saarevet murelahendajad kursuse “Introduction to Programming” jaoks [54] ja 2023. aastal Henri Laats kursuse “Introduction to programming II” jaoks [55]. Selle tulemusena on murelahendajad nüüdseks kasutusel vähemalt kümnel erineval kursusel Tartu Ülikoolis.

## 3.3 Murelahendajate keskkond

Kursusel “Programmeerimise alused II” kasutatavad murelahendajad paiknevad murelahendajate keskkonnas Murelahendaja, mis asub aadressil https://progtugi.cs.ut.ee/ [56]. Murelahendaja on ainult eestikeelne veebirakendus, mis on mõeldud murelahendajate loomiseks ja näitamiseks. Selle lõi Vello Vaherpuu 2016. aastal oma bakalaureusetöö raames [46]. Kasutatavas keskkonnas on kaks vaadet: kasutaja vaade (joonis 6-10) ja külastaja vaade (joonis 11-14).

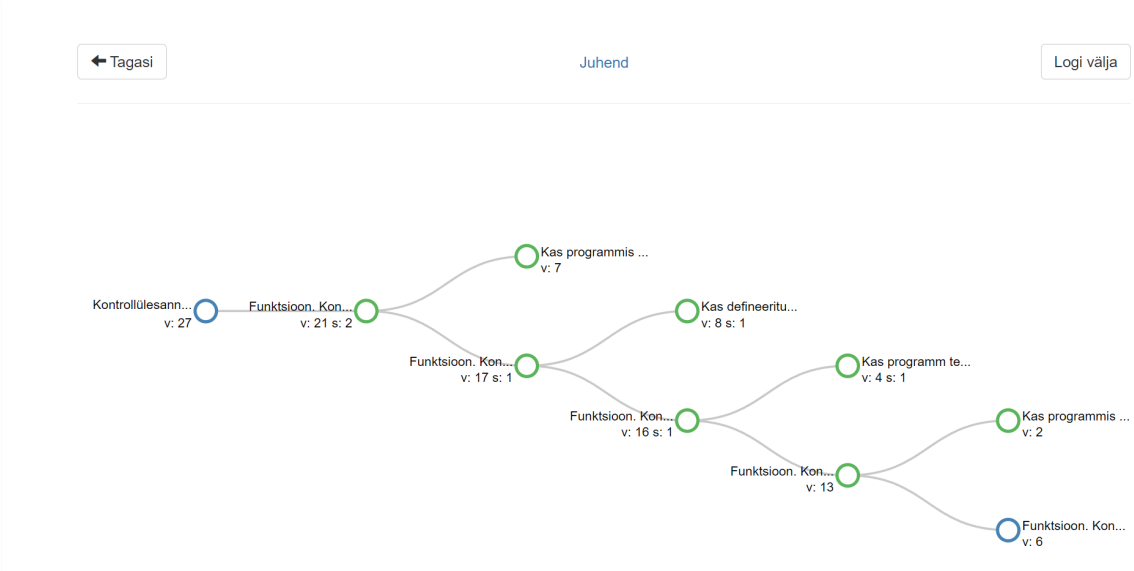
### 3.3.1 Kasutaja vaade

Kasutajaks saamiseks tuleb veebirakenduses registreeruda, luues uue kasutajakonto, või sisse logida, kui kasutajakonto juba on olemas. Kasutaja vaates saab luua ja muuta murelahendajaid, jälgida nende kasutamise statistikat, jagada neid teiste kasutajatega ja saata neid teistele, kasutades loodud linki. Kasutaja vaadet võib nimetada ka administraatori vaateks, kuna see võimaldab hallata kõiki veebirakenduses olemasolevaid murelahendajaid. Lisaks, pärast sisse logimist lisatakse sõna “admin” veebirakenduse lingile, mis kinnitab administratiivset juurdepääsu.



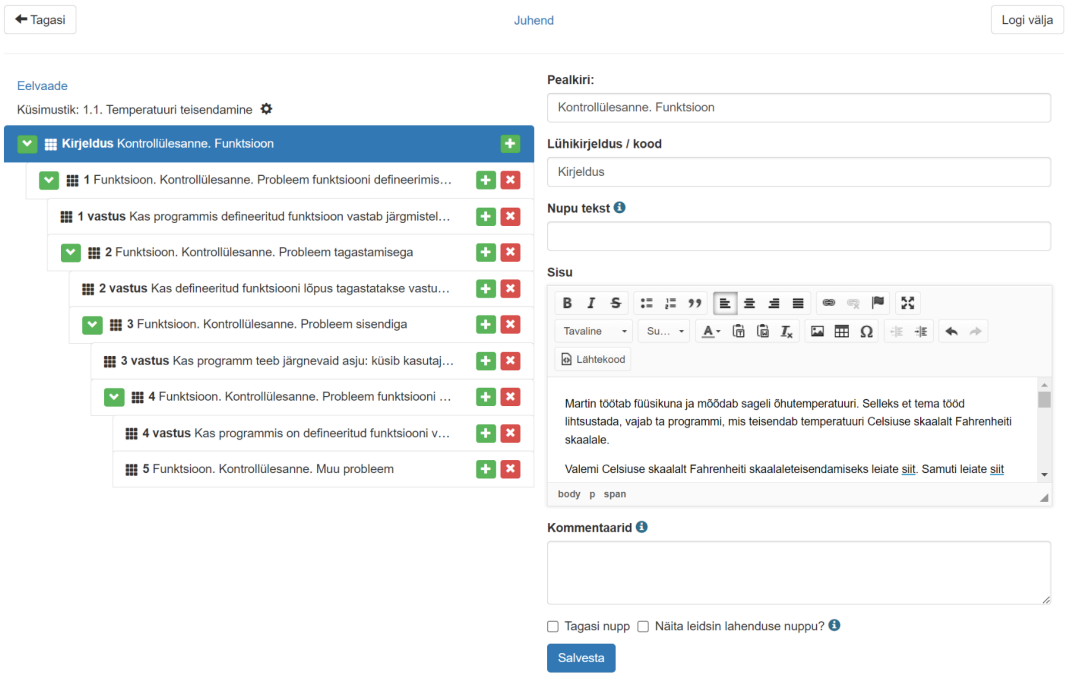

Joonis 6. Kasutaja avalehe vaade.

Pärast sisse logimist suunatakse kasutaja esialgu avalehele (joonis 6). Avalehe ülaosas on horisontaalne navigatsiooniriba, mille keskel on sõna “Juhend”, viidates dokumendile [57], mis sisaldab väikest juhendit veebirakenduse kasutamise kohta. Paremal on nupp, mille abil saab kontost välja logida. Avalehe keskel on nimekiri erinevatest murelahendajatest. Nimekiri koosneb viiest veerust: esimene veerg sisaldab linke murelahendajate nimede kujul (joonis 8); teine veerg sisaldab märksõnu ehk silte murelahendajate esiletõstmiseks; kolmas veerg sisaldab linke murelahendajate statistikale (joonis 7); neljas veerg sisaldab ikoone seadistamiseks, kloonimiseks ja jagamiseks; viies veerg sisaldab linke murelahendajate vaatamiseks (joonis 9). Konkreetse murelahendaja seaded avanevad, kui klõpsata hammasratta ikoonile. Seal on võimalik muuta seadeid nagu murelahendaja nimi, märksõnad, lehe nimi, õnnestumise lehe pealkiri ja nupu tekst. Lehe pealkiri on vahekaardi nimi, mis kuvatakse murelahendaja vaatamisel. Kui pealkiri ei ole määratud, kasutatakse murelahendaja vaikimisi nime. Klõpsates kahe faili ikoonil, kloonitakse murelahendaja. Klõpsates isiku ikoonil, saab murelahendaja jagada teise kasutajaga, sisestades kasutajanime või e-posti aadressi. Lisaks, nimekirja ülaosas on tekst “Filtreeri” koos sisendväljaga, mis võimaldab filtreerida murelahendajaid nime, sildi ning lingi järgi. Seal on ka nupp “Lisa murelahendaja”, mis võimaldab luua uue murelahendaja.



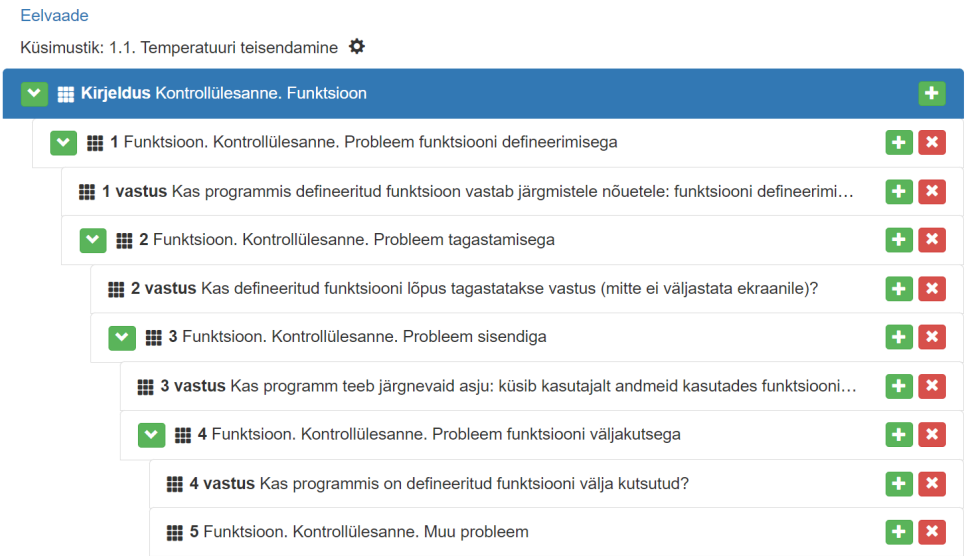

Joonis 7. Murelahendaja statistika vaade.

Lingil “Statistika” vajutades jõuab kasutaja murelahendajate statistikat sisaldavale lehele (joonis 7). Statistikas saab kasutaja ülevaate murelahendaja struktuurist, mis on esitatud ringide kujul sammudena ja hallide joontena, näidates üleminekuid sammude vahel. Murelahenduse struktuur on puu kujul, kus iga küsimus hargneb mitmes suunas. Esimene samm algab vasakul pool ja viimane samm lõpeb paremal pool. Rohelised ringid lehel tähendavad, et sammule on lisatud õnnestumise lehte, samas kui sinised ringid ei sisalda õnnestumise lehte. Statistika lehel kuvatakse iga ringi kõrval selle pealkirja algus ja kogus, mitu korda külastajad on sammu küsimuse või vastusega vaadanud. Sümbol “v” tähendab, mitu korda on külastaja teatud sammu külastanud, ja sümbol “s” tähendab, mitu korda on külastaja ülesande lahendanud, vajutades sammu vastavale nupule, mis sisaldab teksti “Sain korda” ning mis tähendab õnnestunud lahendust. Nende sümbolite väärtusi ei saa pidada kvalitatiivseteks näitajateks. Liikumine järgmistele sammudele suurendab automaatselt eelmiste sammude külastuste arvu. Lisaks, kui üliõpilane lahendab ülesande, võib ta sulgeda vahekaardi, vajutamata nuppu “Sain korda”, mis ei kajastu lahenduste arvu suurenemisena. Lisaks võib sama kasutaja korduvalt vajutada nuppu “Sain korda”, mis kunstlikult suurendab lahenduste arvu. Lisaks väärtustele statistika lehel saab kasutaja liikuda hiirega ringi kohal, et näha konkreetse sammu täieliku pealkirja, ning vajutada ringil, et varjata või paljastada järgmised sammud, mis järgnevad sellele.




Joonis 8. Murelahendaja loomise vaade.

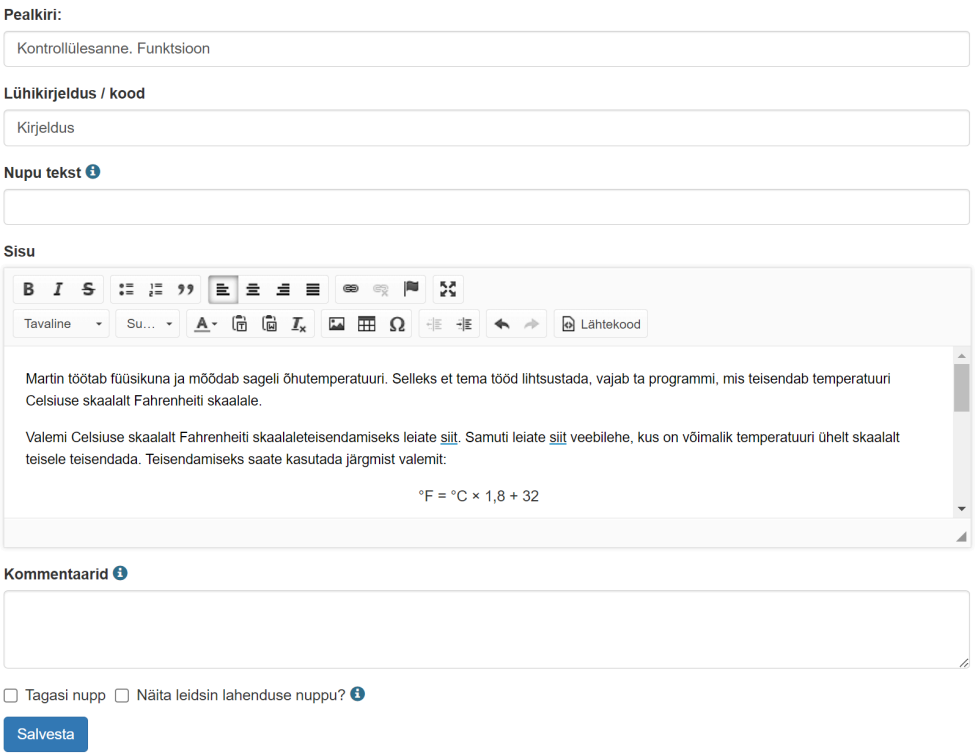
Vajutades avalehel murelahendaja nime kujul oleval lingil, jõuab kasutaja lehele, kus on võimalik murelahendajat redigeerida ja muuta (joonis 8). Lehe vasakul poolel kuvatakse murelahendaja struktuur (joonis 9) ja paremal pool konkreetse sammu sisu (joonis 10).




Joonis 9. Murelahendaja struktuuri vaade.

Murelahendaja struktuur (joonis 9) on esitatud sammude nimekirjana, mis on samuti puu kujul. Sammude nimekirjas on vasakpoolsed taanded, mis näitavad, kui sügavale üks samm on teise suhtes. Igas murelahendajas esitatakse esimeses sammus alati ülesande tekst, millele

järgneb esimene küsimus. Iga küsimuse puhul on kaks võimalust, millest esimene viib vihjete ja abi juurde, mis on küsimusega seotud, ja teine viib järgmise küsimuse juurde. Murelahendaja lõpus on vihje, mis soovitab üliõpilasel pöörduda abi saamiseks kursuse juhendaja poole, kui lahendust ei ole leitud. Murelahendaja struktuuri sammu lisamiseks tuleb vajutada rohelisele pluss-nupule paremal ja sammu eemaldamiseks tuleb vajutada punasele rist-nupule paremal. Konkreetse sammu ja sellega seotud sügavamate elementide lohistamiseks, tuleb kasutada haarde nuppu, hoides seda all ja seejärel vabastades. Lisaks on võimalik kõik sammud konkreetse sammu alla peita, vajutades rohelisele allapoole suunatud noolega nupule.



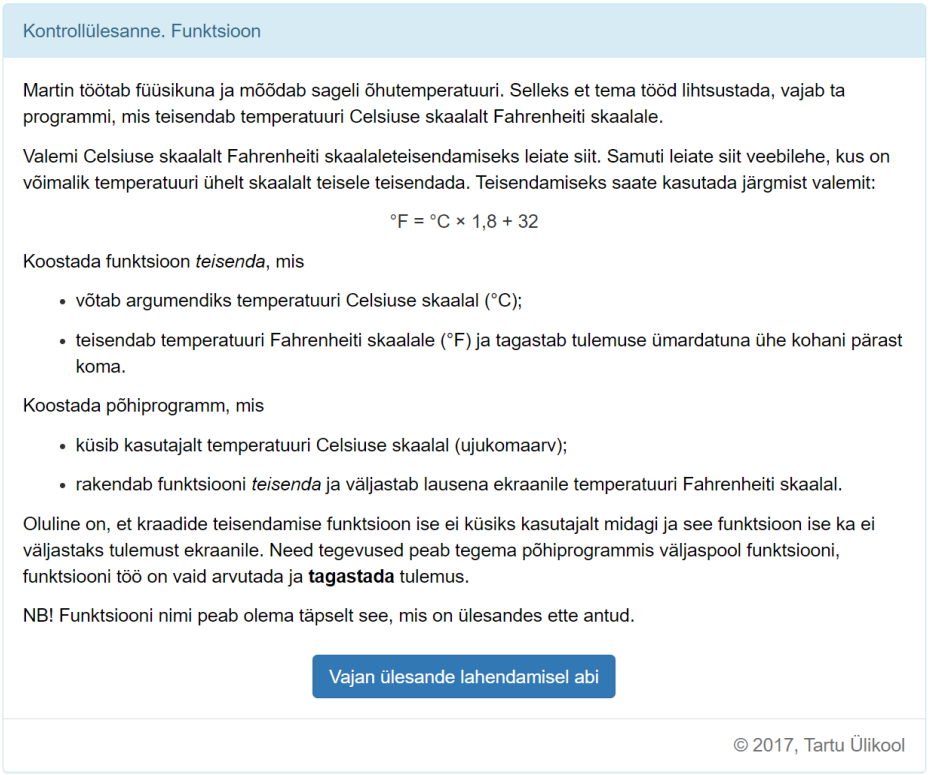

Joonis 10. Murelahendaja sisu vaade.

Murelahendajate sammu sisu (joonis 10) on esitatud vormi kujul, mille kasutaja peaks täitma. Vormis on võimalik kõigepealt koostada pealkiri, sisestades selle sisendväljale “Pealkiri” ning määrata talle sammukood, kasutades sisendvälja “Lühikirjeldus / kood”, mis kuvatakse murelahendaja struktuuris (joonis 9) pealkirjast vasakul pool rasvases kirjas. Sisendväljal “Nupu tekst” saate koostada teksti nupule, mis viib antud sammu juurde. Vormi sisusse saab kirjutada mitmesugust teksti ja seda ka vormindada: panna tekst paksuks, kaldkirjas või üle kriipsutada, luua loendeid, joondada teksti servadesse või keskele, värvida teksti, lisada tekstile linke, lisada pilte või koostada tabeleid ja palju muud. Sisu sees on ka nupp

“Lähtekood”, millele vajutades muutub tekst HTML-koodiks ja seda saab redigeerida. Vorm sisaldab ka kommentaari, kuhu saab kasutaja enda jaoks olulist teavet kirjutada. Valik “Tagasi nupp” lisab sammule lisanupu, mis viib külalise tagasi eelmisele sammule. Valik “Näita leidsin lahenduse nuppu?” lisab sammule lisanupu, mis viib külastaja õnnestumise lehele ja suurendab seega lahendatud ülesannete arvu. Kui muudatused on tehtud, tuleb sammu sisu salvestada ja selleks vajutada nupule “Salvesta”.

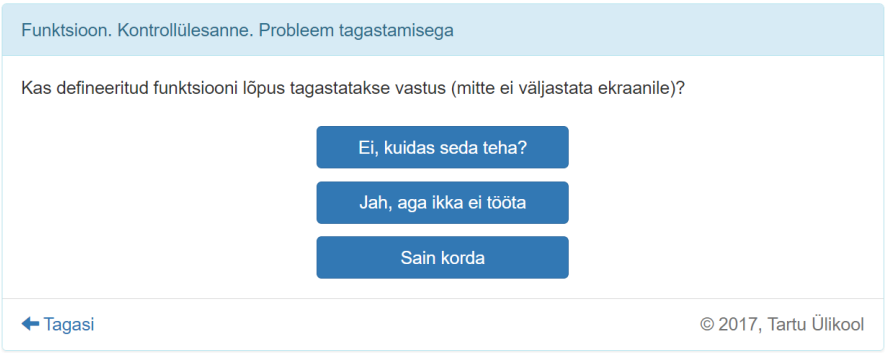
### 3.3.2 Külastaja vaade

Külaliseks saamiseks tuleb järgida kasutaja loodud ja esitatud linki. Külaliseks ei ole vaja luua kasutajakontot. Külalise vaates saab ainult vaadata murelahendajat, klõpsata nuppe sammude vahel liikumiseks ja saada vihjeid ülesande lahendamiseks. Teised tegevused, nagu murelahendaja muutmine või statistika vaatamine, pole külalisele saadaval.



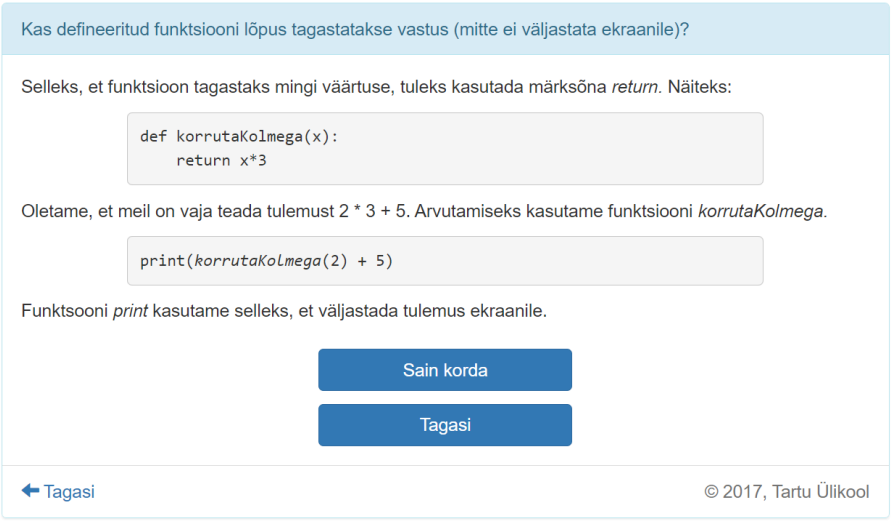

Joonis 11. Külalise vaade avalehele.

Vajutades lingile, jõuab külastaja avalehele (joonis 11), mis on murelahendaja esimene samm. Esimeses sammus kuvatakse külalisele ülesande teksti. Kui ta vajutab nupule “Vajan ülesande lahendamisel abi”, siis suunatakse ta lehele, kus esitatakse esimene murelahendaja küsimus (joonis 12).

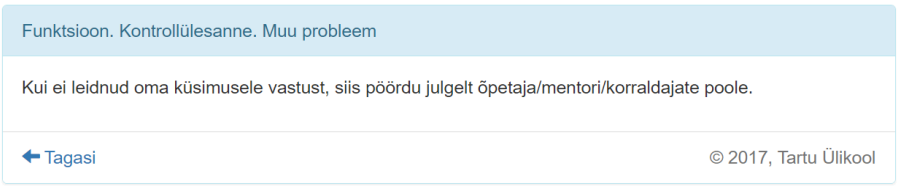



Joonis 12. Murelahendaja küsimuse vaade.

Külalisele esitatakse lehel küsimus (joonis 12) ning tal on kaks valikut. Kui külalisele ei ole küsimusega probleeme, siis tuleb tal vajutada nupule "Jah, aga ikka ei tööta", et liikuda edasi järgmise küsimuse juurde. Kui aga on probleeme, siis tuleb tal vajutada nupule "Ei, kuidas seda teha?", et näha vihjet (joonis 13). Küsimuse ja vihje juures on spetsiaalne nupp "Sain korda", mis näitab lahenduse õnnestumist.




Joonis 13. Murelahendaja vihje vaade.




Joonis 14. Murelahendaja viimase vihje vaade.

Pärast viimast küsimust suunatakse külastaja lehele koos vihjega (joonis 14), kus soovitatakse võtta ühendust kursuse eest vastutava isikuga.

# 4. Metoodika

## 4.1 Ülesannete koostamine

Kursusel “Programmeerimise alused II” on oluline koostada ülesanded, mida üliõpilased peavad igal nädalal sooritama. Enne ülesannete koostamise alustamist oli esmane eesmärk tutvuda kursuse materjalidega. See tagas võtmemõistete põhjaliku mõistmise, mis omakorda võimaldab koostada asjakohaseid ülesandeid, mis vastavad kursuse õppe-eesmärkidele ja toetavad üliõpilaste õppimist. Kursuse materjalide hulka kuuluvad videoloengud Moodle'i keskkonnas [16] ja õppematerjalid kursuse lehel [3]. Lisaks vaadati üle eelmise aasta ülesanded Lahendus keskkonnas [37], et mõista, kuidas ülesanded peaksid välja nägema.

Pärast materjali ja ülesannete ülevaatamist algas ülesannete koostamise protsess. Ülesanded koostati etapiviisiliselt, alustades esimesest õppenädalast ja lõpetades viimase nädalaga. Ülesannete tekstid kirjutati kõigepealt eraldi dokumenti, millest osa kandis kursuse juhendaja seejärel üle Lahendus keskkonda. Lisaks valmistati iga koostatud ülesande jaoks ette funktsioon või programm, et tagada nende rakendatavus. Lisaks vaadati, et uued ülesanded ei oleks kordamise vältimiseks liiga sarnased vanade ülesannetega. Lõpuks kohandati ülesannete tekste mitme nädala jooksul, et parandada vigu ja teha konkreetseid muudatusi.

Lõputöö käigus koostati kokku 39 ülesannet, millest 13 on koduülesanded. Ülejäänud 26 on praktikumi ehk harjutusülesanded. Nende hulgas on viis lisaülesannet, mis on vabatahtlikud ja mida soovitatakse sooritada pärast põhiülesannete lahendamist praktikumis.

## 4.2 Murelahendajate koostamine

Kursuse “Programmeerimise alused II” jaoks koostati kokku 26 murelahendajat: 13 neist tehti kodutööde jaoks (kõik koduülesanded) ning 13 praktikumi ülesannete jaoks. Murelahendajad on olemas kõigi esimese ja teise õppenädala ülesannete kohta. Kõik koduülesanded varustati murelahendajatega, kuna üliõpilased lahendavad neid iseseisvalt ja võivad vajada abi vihjete ja näidiskoodide näol. Murelahendajaid ei loodud 13 ülesande jaoks, mis oli tingitud erinevatest põhjustest. Kõikide praktikumi ülesannete jaoks ei tehtud murelahendajaid, kuna üliõpilased lahendavad neid auditooriumis, kus õppejõud saab neid abistada. Lisaülesanded ei nõua üldiselt murelahendajaid, kuid üks tehti 4. nädala ülesande jaoks, sest see tundus teistest raskem.

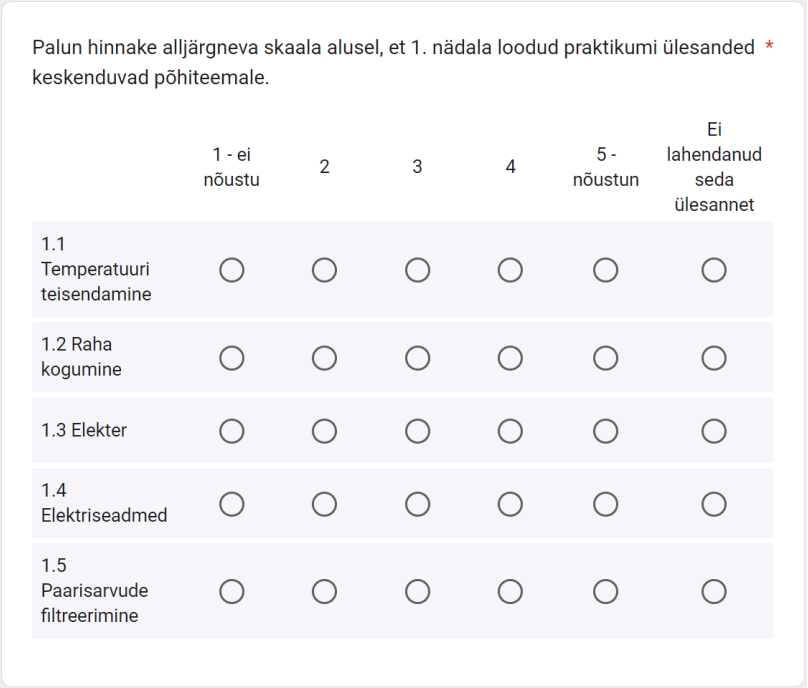
Enne murelahendajate koostamise alustamist vaadati läbi “Programmeerimise alused II” kursuse olemasolevad murelahendajad, et mõista, kuidas need peaksid külalistele välja nägema. Samuti aitab murelahendajate vaatamine mõista, milliste koodiosadega on üliõpilastel kõige rohkem probleeme. Seejärel registreeriti konto Murelahendaja veebirakendusse [56], mis võimaldas alustada murelahendajate koostamist. Murelahendajate loomisel kasutati varasemate näidete põhjal koostatud küsimusi, mis keskendusid koodi osadele, mis on üliõpilaste jaoks eriti keerulised. Vihjete sisu peaks olema piisav, et üliõpilane saaks ülesannet edukalt lahendada, kuid ilma täielikku lahendust pakkumata. Samuti ei tohiks murelahendaja anda lahendust ette. Pärast murelahendaja koostamist kontrolliti külalisena, kas see toimib korrektselt.

## 4.3 Tagasiside küsimine

Selles bakalaureusetöös kasutati koostatud kodu- ja praktikumi ülesannete ning murelahendajate hindamiseks tagasiside küsimustikku. Tagasiside küsimustik saadeti üliõpilatele meili teel. Kuigi tagasiside küsimustiku täitmine oli vabatahtlik, sai üliõpilane selle eest ühe lisapunkti, mis võis olla väike motivaator. Küsimustik keskendus ülesannetele ja murelahendajatele kursuse esimese kahe õppenädala jooksul, et võimaldada tulemuste õigeaegset analüüsi enne lõputöö esitamist. Tagasisidet koguti 18.–28. aprillini 2024.

Tagasiside küsimustiku koostamiseks kasutati Google'i vormi (ingl *Google Forms*) [58], mis on vahend küsitluste, küsimustike, testide ja muude andmete kogumiseks mõeldud vormide loomiseks. Esimesena koostati tagasiside küsimustiku pealkiri ning kirjeldus, kus täpsustati küsimustiku eesmärk, küsimuste arv ja eeldatav kestus minutites. Seejärel tehti oluline muudatus: küsimustik ei olnud anonüümne, et oleks võimalik üliõpilastele anda selle täitmise eest lisapunkt.

Seejärel alustati küsimuste loomist, mis on otseselt seotud kursuse ülesannete ja murelahendajatega. Kuna tagasiside küsimustik piirdus esimese ja teise õppenädalaga, ei olnud võimalik saada täielikku ülevaadet kõigist uutest ülesannetest ja murelahendajatest. Seetõttu otsustati keskenduda eraldi esimese ja teise nädala ülesannetele ja murelahendajatele. Küsimustik koosnes kolmest jaotisest. Esimene jaotis sisaldas küsimustiku kirjeldust ja küsimust, kus üliõpilane pidi sisestama oma nime. Teine jaotis sisaldas küsimusi kursuse esimese õppenädala kohta. Kolmas jaotis sisaldas küsimusi kursuse teise õppenädala kohta.



Palun hinnake alljärgneva skaala alusel, et 1. nädala loodud praktikumi ülesanded * keskenduvad põhiteemale.

| | 1 - ei nõustu | 2 | 3 | 4 | 5 - nõustun | Ei lahendanud seda ülesannet |
|---|---|---|---|---|---|---|
| 1.1 Temperatuuri teisendamine | ○ | ○ | ○ | ○ | ○ | ○ |
| 1.2 Raha kogumine | ○ | ○ | ○ | ○ | ○ | ○ |
| 1.3 Elekter | ○ | ○ | ○ | ○ | ○ | ○ |
| 1.4 Elektriseadmed | ○ | ○ | ○ | ○ | ○ | ○ |
| 1.5 Paarisarvude filtreerimine | ○ | ○ | ○ | ○ | ○ | ○ |

Joonis 15. Tagasiside küsimuse näide.

Aastal 2024 osales kursusel “Programmeerimise alused II” 30 üliõpilast [1]. Tagasiside küsimustikus võib osaleda veidi või oluliselt vähem üliõpilasi, mis mõjutab tulemuste täpsust. Seetõttu otsustati, et kursuse esimese kahe õppenädala küsimused on pigem detailsed, hinnates iga ülesannet erinevate kriteeriumide alusel, mitte kõiki korraga. Näide detailsest küsimusest on esitatud joonisel 15. Igale ülesandele rakendatakse Likert skaalat (ingl *Likert scale*), mis sisaldab kuut valikut: esimesest viiendani on hinnangud, kus üliõpilased väljendavad oma nõusolekut või mitte, ja kuues valik näitab, et üliõpilane ei ole ülesannet lahendanud. Murelahendajate küsimuste puhul on seitse valikut, kus seitsmes tähendab, et ülesanne on lahendatud, kuid murelahendajat ei ole kasutatud. Kokku oli 12 sellist küsimust: kuus teise ja kuus kolmandasse jaotisse. Need küsimused olid kohustuslikud, ilma nendele vastamata ei saanud üliõpilane küsimustikku täita. Lisaks otsustati teise ja kolmanda jaotise lõppu lisada vabatahtlik tekstiküsimus, kus üliõpilased saavad jätta kommentaare oma arvamustest ülesannete ja murelahendajate kohta. Need tekstiküsimused olid vabatahtlikud, ja üliõpilased võisid need vahele jätta, kui nad ei tahtnud vastata.

# 5. Loodud ülesannete ja murelahendajate analüüs

Andmete analüüsimine algas 29. aprillil, päev pärast tagasiside küsimustiku lõppu. Igale üliõpilasele, kes täitis tagasiside küsimustiku, anti lisapunkt. Peale punkti andmist eemaldati andmed, mis olid seotud üliõpilase nimega, et tagada konfidentsiaalsus. Tagasiside küsimustikule vastas 12 üliõpilast, mis moodustab 40% kõigist kursuse osalejatest.

Analüüs algab ülesannete tulemuste hindamisest, jätkub murelahendajate tulemuste analüüsiga ja lõpeb üliõpilaste kommentaaride käsitlusega. Tagasiside küsimustiku tulemused on esitatud tabelites (tabelid 2-13). Tulemused on jaotatud nii, et kõigepealt esitatakse esimese õppenädala andmed, seejärel teise õppenädala andmed. Tagasiside küsimustiku analüüsimisel võeti arvesse ainult need valikud, mille puhul on hinnatud nii ülesandeid kui ka murelahendajaid. Kui vastust ei valinud ükski üliõpilane, siis puudub see valik tabelist. Valikuid, nagu "Ei lahendanud seda ülesannet" või "Lahendasin ülesande, aga ei kasutanud murelahendajat", analüüsis ei kasutata. Samuti arutatakse, kui palju üliõpilasi ligikaudu lahendas konkreetseid ülesandeid ja kui palju kasutas konkreetseid murelahendajaid.

## 5.1 Ülesannete kasutamine ja kasulikkus

### 5.1.1 Esimene nädal

Tabelite 2 ja 4 põhjal lahendasid tagasiside küsimustikus osalenud 11 üliõpilast esimese ja teise ülesande, 9 üliõpilast lahendasid kolmanda ülesande, 8 üliõpilast lahendasid neljanda praktikumi ülesande ning 10 üliõpilast lahendasid viienda praktikumi ülesande. Tabelis 3 ilmnes vastuolu ülesande lahendanud üliõpilaste arvuga: teise ülesande lahendas üks üliõpilane vähem, samas kui neljanda ülesande lahendas üks üliõpilane rohkem. Selgus, et üks üliõpilane ajas segi teise ja neljanda ülesande vastuse valikud teises küsimuses.

Tabelis 2 on esitatud tulemused, mis näitavad, kas esimese nädala praktikumi ülesanded olid põhiteemadele keskendunud. Enamik üliõpilasi nõustus selle väitega, valides iga ülesande kohta kuus või rohkem korda vastuseks "5". Lisaks eelistasid mõned üliõpilased esimese kolme ülesande puhul vastuseks "4" ning teised valisid kõigi ülesannete kohta vastuseks "3". Vastuseid "1" ja "2" ei valinud ükski üliõpilane.

**Tabel 2.** Kas 1. nädala praktikumi ülesanded keskenduvad põhiteemadele.

| | **1.1 Temperatuuri teisendamine** | **1.2 Raha kogumine** | **1.3 Elekter** | **1.4 Elektriseadmed** | **1.5 Paarisarvude filtreerimine** |
|---|---|---|---|---|---|
| 5 (nõustun) | 7 | 9 | 7 | 6 | 8 |
| 4 | 3 | 1 | 1 | 0 | 0 |
| 3 | 1 | 1 | 1 | 2 | 2 |
| Kokku | 11 | 11 | 9 | 8 | 10 |
| Mediaan | 5 | 5 | 5 | 5 | 5 |
| Keskmine hinnang | 4,54 | 4,73 | 4,67 | 4,5 | 4,6 |
| Standardhälve | 0,66 | 0,62 | 0,67 | 0,87 | 0,80 |

Tabelis 3 on esitatud tulemused, mis käsitlevad, kas esimese nädala praktikumi ülesannete selgust. Enamik üliõpilasi nõustus, et ülesanded olid selgelt sõnastatud, valides kuus või enam korda vastuse “5” kõigi ülesannete puhul, välja arvatud kolmas. Kolmanda ülesande puhul oli enamik üliõpilasi osaliselt nõus, valides kuuel korral vastuseks “4”. Lisaks eelistasid mitmed üliõpilased teise ja kahe viimase ülesande kohta vastuseks “3”. Ühtegi vastust “1” või “2” ei valinud keegi.

**Tabel 3.** Kas 1. nädala praktikumi ülesanded on selgelt sõnastatud.

| | **1.1 Temperatuuri teisendamine** | **1.2 Raha kogumine** | **1.3 Elekter** | **1.4 Elektriseadmed** | **1.5 Paarisarvude filtreerimine** |
|---|---|---|---|---|---|
| 5 (nõustun) | 9 | 8 | 3 | 6 | 7 |
| 4 | 2 | 1 | 6 | 2 | 1 |
| 3 | 0 | 1 | 0 | 1 | 2 |
| Kokku | 11 | 10 | 9 | 9 | 10 |
| Mediaan | 5 | 5 | 4 | 5 | 5 |
| Keskmine hinnang | 4,82 | 4,7 | 4,33 | 4,56 | 4,5 |
| Standardhälve | 0,39 | 0,64 | 0,47 | 0,68 | 0,81 |

Tabelis 4 on esitatud tulemused, mis käsitlevad, kas esimese nädala praktikumi ülesannete sobivust raskusastme osas. Enamik üliõpilasi nõustus selle väitega, valides viiel või enamal korral vastuseks “5” iga ülesande puhul. Kuni kaks üliõpilast olid selle väitega osaliselt nõus, valides vastuseks “4”. Lisaks valisid mitmed osalejad esimese nelja ülesande puhul vastuseks “3”, samas kui ainult üks üliõpilane valis viimase, viienda ülesande puhul vastuseks “2”. Ühtegi vastust “1” ei valinud keegi.

**Tabel 4.** Kas 1. nädala praktikumi ülesanded on sobiva raskusastmega.

| | **1.1 Temperatuuri teisendamine** | **1.2 Raha kogumine** | **1.3 Elekter** | **1.4 Elektriseadmed** | **1.5 Paarisarvude filtreerimine** |
|---|---|---|---|---|---|
| 5 (nõustun) | 7 | 8 | 6 | 5 | 7 |
| 4 | 2 | 2 | 2 | 1 | 2 |
| 3 | 2 | 1 | 1 | 2 | 0 |
| 2 | 0 | 0 | 0 | 0 | 1 |
| Kokku | 11 | 11 | 9 | 8 | 10 |
| Mediaan | 5 | 5 | 5 | 5 | 5 |
| Keskmine hinnang | 4,45 | 4,64 | 4,56 | 4,375 | 4,5 |
| Standardhälve | 0,78 | 0,64 | 0,68 | 0,86 | 0,92 |

Vastavalt tulemustele (tabelid 2-4) lahendas enamik üliõpilasi ülesandededukalt, mis näitab, et need olid adekvaatselt koostatud ja vastasid üliõpilaste ettevalmistuse tasemele. Enamik üliõpilasi nõustus kolme väitega, mille täpsem sisu võiks olla välja toodud. Kõigi kriteeriumide osas oli keskmine vastus suurem kui 4, mis oli positiivne tulemus. Lisaks oli standardhälve kõigi kriteeriumide puhul väiksem kui üks, mis tähendab, et enamik väärtusi oli koondunud keskmise lähedale.

### 5.1.2 Teine nädal

Tabelite 6 ja 7 põhjal lahendasid kõik tagasiside küsimustikus osalenud üliõpilased kõik kolm ülesannet. Tabelis 5 tuvastati andmetes vastuolu: kõigis kolmes ülesandes oli üliõpilaste arv ühe võrra väiksem. Selgus, et ühes üliõpilases teise nädala esimeses küsimuses valis kõikjal vastuse, et ta ei lahendanud ülesannet, kuigi kahes järgmises küsimuses valis ta erinevad

vastused. Ei ole selge, kas see valik tehti kogemata või tahtlikult, et tulemusi ebatäpsemaks muuta.

Tabelis 5 on esitatud tulemused, mis käsitlevad, kas teise nädala koduülesannete keskendumist põhiteemadele. Peaaegu kõik üliõpilased nõustusid selle väitega, valides vastuseks “5”. Ainult üks üliõpilane valis teise ülesande puhul vastuseks “4”. Ükski üliõpilane ei valinud vastuseks “3”, “2” ega “1”.

**Tabel 5.** Kas 2. nädala koduülesanded keskenduvad põhiteemadele.

| | **2.1 Kursused** | **2.2 Identsete elementide arv** | **2.3 Värviliste pallide otsimise võistlus** |
|---|---|---|---|
| 5 (nõustun) | 11 | 10 | 11 |
| 4 | 0 | 1 | 0 |
| Kokku | 11 | 11 | 11 |
| Mediaan | 5 | 5 | 5 |
| Keskmine hinnang | 5 | 4,91 | 5 |
| Standardhälve | 0 | 0,29 | 0 |

Tabelis 6 on toodud andmed selle kohta, kui selgelt on sõnastatud teise nädala koduülesanded. Enamik üliõpilasi nõustus väitega, valides teise ülesande puhul kaheksa korda ja esimese ning kolmanda ülesande puhul kümme korda vastuseks “5”. Mõned üliõpilased olid väitega osaliselt nõus, valides teise ülesande puhul kolm korda vastuseks “4” ja esimese ning kolmanda ülesande puhul ühe korda. Lisaks valiti üks neutraalne vastus “3” iga ülesande kohta. Ükski üliõpilane ei valinud vastuseks “1” ja “2”.

**Tabel 6.** Kas 2. nädala koduülesanded on selgelt sõnastatud.

| | **2.1 Kursused** | **2.2 Identsete elementide arv** | **2.3 Värviliste pallide otsimise võistlus** |
|---|---|---|---|
| 5 (nõustun) | 10 | 8 | 10 |
| 4 | 1 | 3 | 1 |
| 3 | 1 | 1 | 1 |
| Kokku | 12 | 12 | 12 |
| Mediaan | 5 | 5 | 5 |
| Keskmine hinnang | 4,75 | 4,58 | 4,75 |
| Standardhälve | 0,60 | 0,64 | 0,60 |

Tabelis 7 on esitatud andmed, mis käsitlevad teise nädala koduülesannete raskusastet. Enamik üliõpilasi nõustus väitega, valides teise ülesande puhul üheksa korda ja esimese ning kolmanda ülesande puhul kümme korda vastuseks “5”. Samuti valisid mõned üliõpilased vastuseks “4” ja “3”. Ükski üliõpilane ei valinud vastuseks “1” ja “2”.

**Tabel 7.** Kas 2. nädala loodud koduülesanded on sobiva raskusastmega.

| | **2.1 Kursused** | **2.2 Identsete elementide arv** | **2.3 Värviliste pallide otsimise võistlus** |
|---|---|---|---|
| 5 (nõustun) | 10 | 9 | 10 |
| 4 | 1 | 1 | 1 |
| 3 | 1 | 2 | 1 |
| Kokku | 12 | 12 | 12 |
| Mediaan | 5 | 5 | 5 |
| Keskmine hinnang | 4,75 | 4,58 | 4,75 |
| Standardhälve | 0,60 | 0,76 | 0,60 |

Tulemuste kohaselt (tabelid 5-7) lahendas enamik üliõpilasi ülesanded, mis näitab, et püstitatud ülesanded olid lahendatavad ja vastasid üliõpilaste ettevalmistuse tasemele. Enamik üliõpilasi nõustus kolme väitega: nad olid rohkem nõus esimese ja kolmanda ülesandega ning veidi vähem nõus teise ülesandega. Kõigi kriteeriumide puhul oli keskmine

vastus suurem kui 4, mis näitab positiivset tulemust. Standardhälve kõigi kriteeriumide puhul oli väiksem kui üks, mis tähendab, et enamik väärtusi andmekogumis on koondunud keskmise väärtuse lähedale.

## 5.2 Murelahendajate kasutamine ja kasulikkus

### 5.2.1 Esimene nädal

Tabelite 9 ja 10 andmete põhjal selgub, et tagasiside küsimustikus osalenud viis üliõpilast kasutasid murelahendajat esimeses ja teises ülesandes, nelja üliõpilast kolmandas ja viiendas ülesandes ning kolme üliõpilast neljandas ülesandes. Tabelis 8 on andmete põhjal vastuolu üliõpilaste arvuga, kes kasutasid murelahendajat neljanda ja viienda ülesande puhul. Täpsemalt, neljanda küsimuse puhul valis üks üliõpilane kaks muud vastust, erinevalt viiendast ja kuuendast küsimusest. Kokkuvõttes, kuna murelahendajat ei kasutanud palju üliõpilasi, võivad tulemused olla ebatäpsed. See võib mõjutada lõpptulemust, mis ei pruugi olla täpne.

Tabelis 8 on esitatud tulemused, mis käsitlevad, kas esimese nädala ülesannete murelahendajate arvestust võimalike raskuskohtadega, mis tekivad ülesannete lahendamisel. Mõned üliõpilased nõustusid väitega, valides vastuseks “5” ja “4”, teised jäid neutraalsed, valides vastuseks “3”, ning mõned olid osaliselt eriarvamusel, valides vastuseks “2”. Ükski üliõpilane ei valinud vastuseks “1”. Murelahendajate keskmine vastus oli umbes 3. Peaaegu kõikide murelahendajate puhul, välja arvatud kolmanda puhul, oli standardhälve suurem kui üks, mis viitab suurele varieeruvusele vastusevariantide vahel.

**Tabel 8.** Kas 1. nädala ülesannete murelahendajad arvestavad võimalike raskuskohtadega, mis tekivad ülesannete lahendamisel.

| | **1.1 Temperatuuri teisendamine** | **1.2 Raha kogumine** | **1.3 Elekter** | **1.4 Elektriseadmed** | **1.5 Paarisarvude filtreerimine** |
|---|---|---|---|---|---|
| 5 (nõustun) | 2 | 2 | 0 | 1 | 2 |
| 4 | 0 | 0 | 1 | 1 | 1 |
| 3 | 2 | 2 | 2 | 1 | 1 |
| 2 | 1 | 1 | 1 | 1 | 1 |
| Kokku | 5 | 5 | 4 | 4 | 5 |
| Mediaan | 3 | 3 | 3 | 3,5 | 4 |
| Keskmine hinnang | 3,6 | 3,6 | 3 | 3,5 | 3,8 |
| Standardhälve | 1,20 | 1,20 | 0,71 | 1,12 | 1,17 |

Tabelis 9 on toodud tulemused, mis näitavad, kuidas esimese nädala ülesannete murelahendajad aitavad lahendada tekkinud probleeme. Mõned üliõpilased valisid vastuseks “5”, “3” ja “2”. Ükski üliõpilane ei valinud vastuseks “4” ja “1”. Esimese, teise ja viienda murelahendaja keskmine vastus oli veidi üle 3, samas kui standardhälve oli suurem kui üks, näidates suurt varieeruvust vastuste. Neljanda ja viienda murelahendaja keskmine hinne jäi alla 3, kuid standardhälve oli väiksem kui üks, mis viitab sellele, et enamik vastuseid koondusid keskmise väärtuse lähedale.

**Tabel 9.** Kas 1. nädala ülesannete murelahendajad aitavad ülesande lahendamisel tekkinud probleeme lahendada.

| | **1.1 Temperatuuri teisendamine** | **1.2 Raha kogumine** | **1.3 Elekter** | **1.4 Elektriseadmed** | **1.5 Paarisarvude filtreerimine** |
|---|---|---|---|---|---|
| 5 (nõustun) | 2 | 2 | 0 | 0 | 1 |
| 3 | 2 | 2 | 2 | 1 | 2 |
| 2 | 1 | 1 | 2 | 2 | 1 |
| Kokku | 5 | 5 | 4 | 3 | 4 |
| Mediaan | 3 | 3 | 2,5 | 2 | 3 |
| Keskmine hinnang | 3,6 | 3,6 | 2,5 | 2,33 | 3,25 |
| Standardhälve | 1,20 | 1,20 | 0,5 | 0,47 | 1,09 |

Tabelis 10 on esitatud tulemused, mis käsitlevad, kas esimese õppenädala ülesannete murelahendajad ei öelnud lahendust ette. Tulemustest selgub, et mõned üliõpilased nõustusid täielikult või osaliselt väitega, valides vastuseks “5” ja “4”. Ülejäänud üliõpilased jäid neutraalseks, valides vastuseks “3”. Ükski üliõpilane ei valinud vastuseks “2” ja “1”. Esimese kolme murelahendaja keskmine vastus oli võrdne või suurem kui 4, samas kui kahe viimase keskmine vastus jäi alla 4. Standardhälve kõigi murelahendajate puhul oli väiksem kui üks.

**Tabel 10.** Kas 1. nädala ülesannete murelahendajad on sellised, et ei ütle lahendust ise ette.

| | **1.1 Temperatuuri teisendamine** | **1.2 Raha kogumine** | **1.3 Elekter** | **1.4 Elektriseadmed** | **1.5 Paarisarvude filtreerimine** |
|---|---|---|---|---|---|
| 5 (nõustun) | 2 | 2 | 2 | 1 | 1 |
| 4 | 1 | 1 | 1 | 0 | 0 |
| 3 | 2 | 2 | 1 | 2 | 3 |
| Kokku | 5 | 5 | 4 | 3 | 4 |
| Mediaan | 4 | 4 | 4,5 | 3 | 3 |
| Keskmine hinnang | 4 | 4 | 4,25 | 3,67 | 3,5 |
| Standardhälve | 0,89 | 0,89 | 0,83 | 0,94 | 0,87 |

Tulemuste põhjal (tabelid 8-10) olid üliõpilased esimese kriteeriumi puhul murelahendajate suhtes enam kui neutraalsed, teise kriteeriumi puhul olid nad veidi nõus, ja kolmanda kriteeriumi puhul täielikult nõus.

### 5.2.2 Teine nädal

Tabelite 11-13 põhjal selgub, et tagasiside küsimustikus osalenud viis üliõpilast kasutasid murelahendajat esimeses, teises ja kolmandas ülesandes.

Tabelis 11 on esitatud tulemused, mis käsitlevad, kas teise nädala koduülesannete murelahendajad arvestavad võimalike raskuskohtadega, mis tekivad ülesannete lahendamisel. Enamik üliõpilasi, kes hindasid murelahendajaid, nõustus väitega, valides vastuseks “5”. Lisaks oli üks üliõpilane ka väitega osaliselt nõus, valides kõigi murelahendajate puhul vastuseks “4”. Ükski üliõpilane ei valinud vastuseks “3”, “2” ja “1”.

**Tabel 11.** Kas 2. nädala koduülesannete murelahendajad arvestavad võimalike raskuskohtadega, mis tekivad ülesannete lahendamisel.

| | **2.1 Kursused** | **2.2 Identsete elementide arv** | **2.3 Värviliste pallide otsimise võistlus** |
|---|---|---|---|
| 5 (nõustun) | 4 | 4 | 4 |
| 4 | 1 | 1 | 1 |
| Kokku | 5 | 5 | 5 |
| Mediaan | 5 | 5 | 5 |
| Keskmine hinnang | 4,8 | 4,8 | 4,8 |
| Standardhälve | 0,40 | 0,40 | 0,40 |

Tabelis 12 on esitatud tulemused, mis käsitlevad, kas teise nädala koduülesannete murelahendajad aitavad lahendada ülesande lahendamisel tekkinud probleeme. Enamik üliõpilasi nõustus väitega, valides vastuseks “5”. Üks üliõpilane oli osaliselt nõus, valides kolmandale murelahendajale vastuseks “4”. Teine üliõpilane valis kolmele murelahendajale vastuseks “3”. Ükski üliõpilane ei valinud vastuseks “2” ja “1”.

**Tabel 12.** Kas 2. nädala koduülesannete murelahendajad aitavad ülesande lahendamisel tekkinud probleeme lahendada.

| | **2.1 Kursused** | **2.2 Identsete elementide arv** | **2.3 Värviliste pallide otsimise võistlus** |
|---|---|---|---|
| 5 (nõustun) | 4 | 4 | 3 |
| 4 | 0 | 0 | 1 |
| 3 | 1 | 1 | 1 |
| Kokku | 5 | 5 | 5 |
| Mediaan | 5 | 5 | 5 |
| Keskmine hinnang | 4,6 | 4,6 | 4,4 |
| Standardhälve | 0,80 | 0,80 | 0,80 |

Tabelis 13 on esitatud tulemused, mis käsitlevad, kas teise nädala koduülesannete murelahendajad on sellised, et ei ütle lahendust ise ette. Enamik üliõpilasi oli väitega nõus, valides vastuseks “5”. Lisaks paar üliõpilast valisid vastuseks “4” ja “3”. Ükski üliõpilane ei valinud vastuseks “2” ja “1”.

**Tabel 13.** Kas 2. nädala koduülesannete murelahendajad on sellised, et ei ütle lahendust ise ette.

| | **2.1 Kursused** | **2.2 Identsete elementide arv** | **2.3 Värviliste pallide otsimise võistlus** |
|---|---|---|---|
| 5 (nõustun) | 3 | 4 | 3 |
| 4 | 1 | 0 | 1 |
| 3 | 1 | 1 | 1 |
| Kokku | 5 | 5 | 5 |
| Mediaan | 5 | 5 | 5 |
| Keskmine hinnang | 4,4 | 4,6 | 4,4 |
| Standardhälve | 0,80 | 0,80 | 0,80 |

Tulemuste põhjal (tabelid 11-13) enamik üliõpilasi oli nõus kolme kriteeriumiga. Kõikide murelahendajate kriteeriumide puhul keskmine vastus oli suurem kui 4, mis on positiivne

tulemus. Lisaks, standardhälve oli väiksem kui üks, mis tähendab, et enamik väärtusi andmekogumis on keskmisele väärtusele lähedal hajutatud.

## 5.3 Osalejate kommentaarid

Tagasiside küsimustikus osaledes jätsid mitmed üliõpilased kommentaare kursuse esimese ja teise õppenädala kohta. Kaks üliõpilast jätsid mõlema nädala kohta ühe kommentaari ning ülejäänud kaks jätsid kommentaarid ainult teise õppenädala kohta. Iga kommentaari juurde on lisatud tagasiside küsimustikus osaleja number.

Kursuse 1. õppenädala kommentaarid on järgmised:

“Ülesanded hästi koostatud ja toetavad ka õppijat.” - Osaleja 1

“kõik on ok” - Osaleja 2

Kursuse 2. õppenädala kommentaarid on järgmised:

“Toetavad ja kinnistavad ülesanded. Hea, et ka praktikumis koos ülesandeid lahendame.” - Osaleja 1

“👍🏻” - Osaleja 2

“2.2 oli väga lühike, aga eks õppimise seisukohast ongi ilmselt nii mõistlikum.” - Osaleja 3

“Mulle meeldis, et sai rohkem keskenduda teemale, kui sellele, kuidas andmete lugemist failist lahendada” - Osaleja 4

Enamik kommentaare, mida jätsid esimene, teine ja neljas osaleja, olid positiivsed. Üliõpilased väljendasid head suhtumist nii ülesannetesse kui ka murelahendajatesse. Kolmas osaleja märkis, et teise õppenädala teine ülesanne tundus liiga lühike, kuid ei pidanud seda suureks probleemiks. Võimalik, et see mulje oli tingitud asjaolust, et ülesandes ei nõutud kasutaja sisendit ega failist lugemist, mis tavaliselt suurendab koodi kirjutamise mahtu. Selle olukorra parandamiseks võiks järgmisel korral vahetada kaks esimest ülesannet omavahel, nii et lühem on enne ja siis järgneb ülesanne, mis hõlmab ka kasutaja sisendit ja failist lugemist.

## Kokkuvõte

Bakalaureusetöö eesmärk oli luua programmeerimise kursuse “Programmeerimise alused II” uued ülesanded ning nende lahendamist toetavad murelahendajad. Lõputöö teoreetilises osas vaadeldi esmalt kursust ennast, selle materjale, ülesandeid ja murelahendajaid üldiselt ning nende kasutuskeskkonda. Seejärel uuriti põhjalikumalt enamiku ülesannete struktuuri. Praktilises osas koostati kokku 39 ülesannet, millest 13 on koduülesanded ja 26 on praktikumi ülesanded. Samuti loodi 26 murelahendajat. Koostati ka tagasiside küsimustik ning koguti kursuse üliõpilastelt tagasisidet, et hinnata kursuse esimese ja teise õppenädala ülesannete ja murelahendajate kvaliteeti ja kasulikkust.

Tagasiside küsimustikule vastas 12 üliõpilast, mis moodustab 40% osalejatest. Tulemuste põhjal suhtusid üliõpilased positiivselt esimese ja teise õppenädala ülesannetesse, kusjuures enamik neist oli väidetega täielikult nõus. Küsimustikus osalenud üliõpilastest kasutas maksimaalselt 5 üliõpilast vähemalt ühte loodud murelahendajat ülesande lahendamiseks. Väikese kasutajate arvu tõttu murelahendajate puhul ei saa tulemusi pidada täpseteks. Siiski näitavad saadud tulemused, et suurem osa üliõpilastest nõustus täielikult kolme väitega kursuse teise õppenädala murelahendajate kvaliteedi kohta, samas kui esimese õppenädala murelahendajate osas nõustusid nad väidetega vähem. Üliõpilaste kommentaarid olid peamiselt positiivsed.

Edasiarendusena, arvestades üliõpilaste tagasisidet, tasub muuta kursuse “Programmeerimise alused II” esimese õppenädala murelahendajaid. Samuti oleks mõistlik vahetada kursuse teise õppenädala esimene ja teine ülesanne, kuna esimene on mahukam kui teine. Vajadusel saab koostatud ülesandeid ja murelahendajaid ka edaspidi muuta, et neid täiustada. Üliõpilastelt saadud positiivne tagasiside esimese ja teise õppenädala ülesannetele ning teise õppenädala murelahendajatele viitab sellele, et nende arendamine ja kasutamine teistel kursustel võiks pakkuda veelgi rohkem tuge.

## Viidatud kirjandus


[1] Tartu Ülikooli õppeinfosüsteem 2. https://ois2.ut.ee/ (22.11.2023)

[2] Kursuse "Programmeerimise alused" koduleht. https://courses.cs.ut.ee/2023/prog-alused/spring (23.11.2023)

[3] Kursuse "Programmeerimise alused II" koduleht. https://courses.cs.ut.ee/2023/progalused2/spring (23.11.2023)

[4] e-Teatmik: IT ja sidetehnika seletav sõnaraamat. http://www.vallaste.ee/ (29.11.2023)

[5] AKIT: Andmekaitse ja infoturbe portaal. https://akit.cyber.ee/ (29.11.2023)

[6] Pythoni dokumentatsioon. History and License. https://docs.python.org/3/license.html (25.11.2023)

[7] Pythoni koduleht. Downloads. https://www.python.org/downloads/ (15.03.2024)

[8] TIOBE Programming Community index. https://www.tiobe.com/tiobe-index/ (19.04.2024)

[9] PYPL PopularitY of Programming Language Index. https://pypl.github.io/PYPL.html (19.04.2024)

[10] Pythoni koduleht. https://www.python.org/ (29.11.2023)

[11] Thonny, Python IDE for beginners. https://thonny.org/ (26.11.2023)

[12] Thonny Github repository. https://github.com/thonny/thonny (26.11.2023)

[13] Annamaa, A., 2015, Introducing Thonny, a Python IDE for learning programming, *Proceedings of the 15th Koli Calling Conference on Computing Education Research*, 117–121. https://dl.acm.org/doi/10.1145/2828959.2828969 (26.11.2023)

[14] Raspberry Pi Foundation. Install Thonny. https://projects.raspberrypi.org/en/projects/getting-started-with-the-pico/2 (26.11.2023)

[15] Kursuse "Programmeerimine" koduleht. https://courses.cs.ut.ee/2023/programmeerimine/fall (26.11.2023)

[16] Moodle'i koduleht. https://moodle.org/ (30.11.2023)

[17] Digipädevus sõnastik. https://digipadevus.ee/sonastik/ (26.02.2024)

[18] Tartu Ülikooli e-õppe ajakiri. Lähiõppest hajaõppeni. Digiõppe täpsustatud terminid. https://etu.ut.ee/2021/digioppeterminid/ (26.02.2024)

[19] Garrison, D. R., Kanuka, H., 2004, Blended learning: Uncovering its transformative potential in higher education, *The internet and higher education*, 7(2), 95–105. https://doi.org/10.1016/j.iheduc.2004.02.001 (03.04.2024)

[20] López-Pérez, M. V., Pérez-López, M. C., Rodríguez-Ariza, L., 2011, Blended learning in higher education: Students' perceptions and their relation to outcomes, *Computers & education*, 56(3), 818–826. https://doi.org/10.1016/j.compedu.2010.10.023 (03.04.2024)

[21] Staker, H., Horn, M. B., 2012, Classifying K–12 blended learning, *Innosight Institute*. http://192.248.16.117:8080/research/handle/70130/5105 (03.04.2024)

[22] Harvard University Derek Bok Center for Teaching & Learning. Flipped Classrooms. https://bokcenter.harvard.edu/flipped-classrooms (03.04.2024)

[23] Baker, J. W., 2000, The "Classroom Flip": Using Web Course Management Tools to Become the Guide by the Side, *11th International Conference on College Teaching and Learning*, 9–17. https://upcea.edu/wp-content/uploads/2020/09/The-Classroom-Flip-Baker.pdf (03.04.2024)

[24] Lage, M. J., Platt, G. J., Treglia, M., 2000, Inverting the classroom: A gateway to creating an inclusive learning environment, *The journal of economic education*, 31(1), 30–43. https://www.tandfonline.com/doi/epdf/10.1080/00220480009596759 (03.04.2024)

[25] Bishop, J., Verleger, M. A., 2013, The flipped classroom: A survey of the research, *2013 ASEE Annual Conference & Exposition*, 1–10. https://peer.asee.org/22585 (03.04.2024)

[26] Al-Samarraie, H., Shamsuddin, A., Alzahrani, A. I., 2020, A flipped classroom model in higher education: a review of the evidence across disciplines, *Educational Technology Research and Development*, 68(3), 1017–1051. https://doi.org/10.1007/s11423-019-09718-8 (03.04.2024)

[27] Rosiene, C.P., Rosiene, J.A., 2015, Flipping a programming course: The good, the bad, and the ugly, *IEEE Frontiers in Education Conference (FIE)*, 1–3. https://ieeexplore.ieee.org/document/7344151 (03.04.2024)

[28] Lepp, M., Tõnisson, E., 2015, Integrating Flipped Classroom Approach and Work in Pairs into Workshops in Programming Course, *International Conference on Frontiers in Education: Computer Science and Computer Engineering*, 220–226. https://www.researchgate.net/publication/282002190_Integrating_Flipped_Classroom_Approach_and_Work_in_Pairs_into_Workshops_in_Programming_Course (03.04.2024)

[29] Kursuse "Objektorienteeritud programmeerimine" koduleht. https://courses.cs.ut.ee/2024/OOP/spring (04.04.2024)

[30] Weinman, N., 2022, Designing Exercises to Teach Programming Patterns, Electrical Engineering and Computer Sciences University of California. https://digitalassets.lib.berkeley.edu/techreports/ucb/incoming/EECS-2022-257.pdf (13.03.2024)

[31] VanDeGrift, T., 2015, Supporting Creativity and User Interaction in CS 1 Homework Assignments, *Proceedings of the 15th Koli Calling Conference on Computing Education Research*, 54–59. https://dl.acm.org/doi/10.1145/2676723.2677250 (13.03.2024)

[32] Edgcomb, A., Vahid, F., Lysecky, R., Lysecky, S., 2017, Getting Students to Earnestly Do Reading, Studying, and Homework in an Introductory Programming Class, *Proceedings of the 17th Koli Calling Conference on Computing Education Research*, 171–176. https://dl.acm.org/doi/10.1145/3017680.3017732 (13.03.2024)

[33] Epstein, J. L., Van Voorhis, F. L., 2010, More Than Minutes: Teachers' Roles in Designing Homework, *Taylor and Francis Online*, 36(3), 181–193. https://www.tandfonline.com/doi/abs/10.1207/S15326985EP3603_4 (13.03.2024)

[34] Sullivan, M., Sequeira, P., 1996, The impact of purposeful homework on learning, *The Clearing House*, 69, 346–348. https://www.tandfonline.com/doi/pdf/10.1080/00098655.1996.10114337 (13.03.2024)

[35] Cooper, H., 1989, Synthesis of research on homework, *Educational Leadership*, 46(2), 85–91. https://files.ascd.org/staticfiles/ascd/pdf/journals/ed_lead/el198911_cooper.pdf (13.03.2024)

[36] Huang, B., Hew, K. F. T., 2016, Measuring learners' motivation level in massive open online courses, *International Journal of Information and Education Technology*, 6(10), 759–764. https://www.ijiet.org/show-78-893-1.html (13.03.2024)

[37] Lahenduse keskkond. https://lahendus.ut.ee/ (04.03.2024)

[38] Lahenduse kasutustingimused. https://docs.google.com/document/d/1dk1Pp3hXJEX7HllQFdMFo5AXhgzy4zhZv3Qt6-xI_CI/edit#heading=h.df01thmxe31v (04.03.2024)

[39] Discordi koduleht. https://discord.com/ (04.03.2024)

[40] Giurgiu, L., Gligorea, I., 2017, Responsive web design techniques, *International conference knowledge-based organization*, 23(3), 37–42. https://sciendo.com/article/10.1515/kbo-2017-0153 (04.03.2024)

[41] Digiriigi Akadeemia. Kursuse loomise ABC kursus. https://digiriigiakadeemia.ee/ (10.03.2024)

[42] Chandrasekar, R., Srinivas, B., 1997, Automatic induction of rules for text simplification, *Knowledge-Based Systems*, 10(3), 183–190. https://doi.org/10.1016/S0950-7051(97)00029-4 (10.03.2024)

[43] W3Schools Online Web Tutorials. CSS Web Safe Fonts. https://www.w3schools.com/cssref/css_websafe_fonts.php (18.03.2024)

[44] Google fonts. Roboto. https://fonts.google.com/specimen/Roboto (18.03.2024)

[45] Jonassen, D. H., Hung, W., 2006, Learning to troubleshoot: A new theory-based design architecture, *Educational Psychology Review*, 18(1), 77–114. https://link.springer.com/article/10.1007/s10648-006-9001-8 (11.03.2024)

[46] Vaherpuu, V., 2026, Murelahendajate loomise keskkond, TÜ arvutiteaduse instituudi bakalaureusetöö. https://dspace.ut.ee/items/d34d5c0a-3cdb-4f80-bb62-e9d704207a74 (11.03.2024)

[47] Lepp, M., Palts, T., Luik, P., Papli, K., Suviste, R., Säde, M., Hollo, K., Vaherpuu, V., Tõnisson, E., 2018, Troubleshooters for tasks of introductory programming MOOCs, *International Review of Research in Open and Distributed Learning*, 19(4). https://www.irrodl.org/index.php/irrodl/article/download/3639/4728?inline=1 (11.03.2024)

[48] Tartu Ülikooli kursus „Programmeerimisest maalähedaselt''. https://courses.cs.ut.ee/2024/progmaa/spring (11.03.2024)

[49] Pent, M., 2020, Murelahendajad programmeerimise kursusel, TÜ arvutiteaduste instituudi bakalaureusetöö. https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=69770 (11.03.2024)

[50] Klaanberg, A., 2020, Murelahendajate koostamine Tartu Ülikooli kursuse „Objektorienteeritud programmeerimine" tarbeks, TÜ arvutiteaduste instituudi bakalaureusetöö. https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=69730 (11.03.2024)

[51] Kaimre, J., 2021, Murelahendajate koostamine Tartu Ülikooli kursuse „Programmeerimine" jaoks, TÜ arvutiteaduste instituudi bakalaureusetöö. https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=71614 (11.03.2024)

[52] Kivilaan, K., 2022, Murelahendajate koostamine Tartu Ülikooli kursuse „Sissejuhatus andmebaasidesse" jaoks, TÜ arvutiteaduste instituudi bakalaureusetöö. https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=74382 (11.03.2024)

[53] Ott, K. E., 2023, Murelahendajate koostamine Tartu Ülikooli kursuse "Andmebaasid" jaoks ja nende kasutamise analüüs, TÜ arvutiteaduste instituudi bakalaureusetöö. https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=77545 (11.03.2024)

[54] Saarevet, T., 2022, Koduülesannete ja murelahendajate loomine kursusele „Introduction to Programming", TÜ arvutiteaduste instituudi bakalaureusetöö. https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=74534 (11.03.2024)

[55] Laats, H., 2023, Uue ülesandekogu ja kodutöödele vastavate murelahendajate koostamine kursusele „Introduction to programming II'', TÜ arvutiteaduste instituudi bakalaureusetöö. https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=77265 (11.03.2024)

[56] TÜ Troubleshooting. https://progtugi.cs.ut.ee/ (13.03.2024)

[57] Murelahendaja juhend. https://docs.google.com/document/d/12G5_BN_Re26LT7S7Tipf00_L_nZMEFGLWcq1tmlUqjk/edit (18.03.2024)

[58] Google Forms. Online form creator. https://www.google.com/forms/about/ (24.03.2024)

# Lisad

## I Koostatud ülesanded

### Nädal 1. Harjutusülesanne 1.1 Temperatuuri teisendamine

Martin töötab füüsikuna ja mõõdab sageli õhutemperatuuri. Selleks et tema tööd lihtsustada, vajab ta programmi, mis teisendab temperatuuri Celsiuse skaalalt Fahrenheiti skaalale.

Täpsemalt saab teisendamise kohta lugeda siit või katsetada temperatuuri kalkulaatorit. Teisendamiseks saate kasutada järgmist valemit:

$$°F = °C \times 1{,}8 + 32$$

Koostada funktsioon **teisenda**, mis

- võtab argumendiks temperatuuri Celsiuse skaalal (°C);
- teisendab temperatuuri Fahrenheiti skaalale (°F) ja tagastab tulemuse ümardatuna ühe kohani peale koma.

Näited funktsiooni tööst

```
>>> teisenda(7.7)
45.9
>>> teisenda(8.5)
47.3
>>> teisenda(12)
53.6
```

Koostada põhiprogramm, mis

- küsib kasutajalt temperatuuri Celsiuse skaalal (ujukomaarv);
- rakendab funktsiooni **teisenda** ja väljastab lausena ekraanile temperatuuri Fahrenheiti skaalal.

Oluline on, et kraadide teisendamise funktsioon ise ei küsiks kasutajalt midagi ja see funktsioon ise ka ei väljastaks tulemust ekraanile. Need tegevused peab tegema põhiprogrammis väljaspool funktsiooni, funktsiooni töö on vaid arvutada ja tagastada tulemus.

Näited programmi tööst

```
>>> %Run lahendus.py
 Sisestage temperatuur Celsiuse skaalal: 10.5
 Temperatuur Fahrenheiti skaalal on 50.9

>>> %Run lahendus.py
  Sisestage temperatuur Celsiuse skaalal: 12
  Temperatuur Fahrenheiti skaalal on 53.6

>>> %Run lahendus.py
  Sisestage temperatuur Celsiuse skaalal: 35.5
  Temperatuur Fahrenheiti skaalal on 95.9
```

Kui olete juba hulk aega proovinud ülesannet iseseisvalt lahendada ja see ikka ei õnnestu, siis võib-olla saate abi murelahendajalt. Püütud on tüüpilisemaid probleemseid kohti selgitada ja anda vihjeid.

## Nädal 1. Harjutusülesanne 1.2 Raha kogumine

Stanislav on seadnud endale suure eesmärgi, mille saavutamiseks peab ta koguma 5000 eurot. Ta vaatab, kui palju ta keskmiselt kuus teenib ja mitu kuud on projekti elluviimiseni jäänud. Ta otsustab luua programmi, mille abil saab ta kontrollida, kas teatud aja jooksul teenitavast rahast piisab eesmärgiks seatud summa kogumiseks.

Koostada funktsioon **veel_vaja**, mis

- võtab argumentideks kuus teenitud raha eurodes ja kuude arvu;
- tagastab eesmärgi saavutamiseks vajamineva summa eurodes (kui eesmärk on saavutatud, tagastab funktsioon 0).

Näited funktsiooni tööst

```
>>> veel_vaja(720, 5)
1400
>>> veel_vaja(650, 8)
0
>>> veel_vaja(1240, 4)
40
>>> veel_vaja(920, 5)
400
```

Koostada põhiprogramm, mis

- küsib kasutajalt
    - kuus teenitud raha eurodes (täisarv);
    - kuude arvu (täisarv);
- leiab funktsiooni **veel_vaja** kasutades, kui palju raha on vaja veel koguda eesmärgi saavutamiseks, ning väljastab selle lausena ekraanile.
    - Kui eesmärk on saavutatud, väljastab programm ekraanile teate, et olete juba saavutanud või ületanud eesmärgi.

Näited programmi tööst

```
>>> %Run lahendus.py
  Sisestage igakuine summa: 1240
  Sisestage kuude arv: 4
  Teil on vaja koguda veel 40 eurot, et saavutada eesmärk.

>>> %Run lahendus.py
  Sisestage igakuine summa: 860
  Sisestage kuude arv: 8
  Te olete juba saavutanud või ületanud eesmärgi
```

Kui olete juba hulk aega proovinud ülesannet iseseisvalt lahendada ja see ikka ei õnnestu, siis võib-olla saate abi murelahendajalt. Püütud on tüüpilisemaid probleemseid kohti selgitada ja anda vihjeid.

## Nädal 1. Harjutusülesanne 1.3 Elekter

Riigilõiv iga kehtivusaasta eest sõltub jaotusvõrgu kaudu võrguteenuse osutamisest aastase teenuste mahuga:

- kui aastane teenuste maht on kuni 5 GWh, siis tuleb aastas tasuda 960 eurot;
- kui aastane teenuste maht on üle 5 GWh ja mitte rohkem kui 20 GWh, siis tuleb aastas tasuda 1920 eurot;
- kui aastane teenuste maht on üle 20 GWh ja mitte rohkem kui 50 GWh, siis tuleb aastas tasuda 4480 eurot;
- kui aastane teenuste maht on üle 50 GWh, siis tuleb aastas tasuda 6400 eurot.

*Ülesande jaoks on tegelikke kriteeriume lihtsustatud. Võrguteenuste kohta saab lähemalt lugeda siit.

GWh tähistab gigavatt-tundi ning see on energiahulk, mis võrdub ühe miljoni kilovatt-tunniga. Kilovatt-tund (tähis kWh) on energia mõõtühik, mida kasutatakse peamiselt elektri- ja soojusenergeetikas. Üks kilovatt-tund on energiahulk, mida tarbib ning toodab ühtlasel 1-kilovatisel võimsusel töötav seade ühe tunni jooksul.

Koostada funktsioon **aastatasu**, mis

- võtab argumendiks aastase teenuste mahu (GWh);
- tagastab aastatasu.

Näited funktsiooni tööst

```
>>> aastatasu(2)
960
>>> aastatasu(5)
960
>>> aastatasu(10)
1920
>>> aastatasu(25)
4480
>>> aastatasu(75)
6400
```

Aastased teenuste mahud (GWh) on tekstifailis paigutatud eraldi ridadele.

Koostada programm, mis

- küsib kasutajalt faili nime;
- loeb failist aastased teenuste mahud (ujukomaarvud);
- väljastab ekraanile aastased teenuste mahud ja iga taseme kohta aastatasu, rakendades funktsiooni **aastatasu**;
- leiab ja väljastab ekraanile aastatasude kogusumma.

Näited programmi tööst

Faili **aastateenused.txt** sisu:

```
4.3
10.2
26.3
54.7
2.4
5.7
12.2
```

```
>>> %Run lahendus.py
Sisestage failinimi: aastateenused.txt
Aastane teenuste maht: 4.3 GWh - aastatasu: 960 eurot.
Aastane teenuste maht: 10.2 GWh - aastatasu: 1920 eurot.
Aastane teenuste maht: 26.3 GWh - aastatasu: 4480 eurot.
```

Aastane teenuste maht: 54.7 GWh - aastatasu: 6400 eurot.
Aastane teenuste maht: 2.4 GWh - aastatasu: 960 eurot.
Aastane teenuste maht: 5.7 GWh - aastatasu: 1920 eurot.
Aastane teenuste maht: 12.2 GWh - aastatasu: 1920 eurot.
Aastatasude kogusumma: 18560 eurot.

Faili **aastateenused.txt** sisu:
6.3
11.1
13.2
54.7
4.2

>>> %Run lahendus.py
Sisestage failinimi: aastateenused.txt
Aastane teenuste maht: 6.3 GWh - aastatasu: 1920 eurot.
Aastane teenuste maht: 11.1 GWh - aastatasu: 1920 eurot.
Aastane teenuste maht: 13.2 GWh - aastatasu: 1920 eurot.
Aastane teenuste maht: 54.7 GWh - aastatasu: 6400 eurot.
Aastane teenuste maht: 4.2 GWh - aastatasu: 960 eurot.
Aastatasude kogusumma: 13120 eurot.

Kui olete juba hulk aega proovinud ülesannet iseseisvalt lahendada ja see ikka ei õnnestu, siis võib-olla saate abi murelahendajalt. Püütud on tüüpilisemaid probleemseid kohti selgitada ja anda vihjeid.

## Nädal 1. Harjutusülesanne 1.4 Elektriseadmed

Vastavalt energiatõhususe standardile jagunevad elektriseadmed kolme klassi:

- A-klass — tarbib vähem kui 10% maksimaalsest energiast;
- B-klass — tarbib vähemalt 10% maksimaalsest energiast;
- C-klass — tarbib vähemalt 25% maksimaalsest energiast.

Leiti hulk elektriseadmeid, mille klassi polnud võimalik muul viisil kindlaks teha kui testides. Testimisel kasutati 5000 kWh energiat ja leiti, kui mitu kWh energiat need seadmed tarbivad. Näiteks kui elektriseade tarbib 250 kWh energiat, mis on 5%, siis elektriseade kuulub A-klassi. Kui elektriseade tarbib 500 kWh energiat, mis on 10%, siis elektriseade kuulub B-klassi.

Koostada funktsioon **klass**, mis

- võtab argumendiks tarbitud energia koguse kilovatt-tundides (kWh);
- tagastab klassi (sõne A-klass, B-klass või C-klass), arvestades ülaltoodud kriteeriume. Arvutamisel võib abi olla valemist `100 * tarbitud_energia_kogus / 5000`.

Näited funktsiooni tööst

```
>>> klass(100)
'A-klass'
>>> klass(500)
'B-klass'
>>> klass(1000)
'B-klass'
>>> klass(3000)
'C-klass'
```

Koostada programm, mis

- küsib kasutajalt faili nime;
- loeb sellest failist tarbitud energiakogused kilovatt-tundides (täisarvud);
- loendab (rakendades ka funktsiooni **klass**), mitu elektriseadet kuulub A-klassi või B-klassi;
- väljastab tulemuse näites toodud lausena.

Näide programmi tööst

Faili **energiakogused.txt** sisu:

```
500
1000
1250
1245
3000

>>> %Run lahendus.py
  Sisestage failinimi: energiakogused.txt
  A-klassi ja B-klassi elektriseadmeid oli kokku 3.
```

Faili **energiakogused.txt** sisu:

```
424
940
2328
880
370
1940

>>> %Run lahendus.py
  Sisestage failinimi: energiakogused.txt
  A-klassi ja B-klassi elektriseadmeid oli kokku 4.
```

Kui olete juba hulk aega proovinud ülesannet iseseisvalt lahendada ja see ikka ei õnnestu, siis võib-olla saate abi murelahendajalt. Püütud on tüüpilisemaid probleemseid kohti selgitada ja anda vihjeid.

### Nädal 1. Harjutusülesanne 1.5 Paarisarvude filtreerimine

On antud järjend, mis koosneb erinevatest täisarvulistest väärtustest.

Koostada funktsioon **paarisarvude_filter**, mis

- võtab argumendiks täisarvude järjendi;
- tagastab uue järjendi, mis sisaldab ainult algse järjendi paarisarve.

Näited funktsiooni tööst

```
>>> paarisarvude_filter([1])
[]
>>> paarisarvude_filter([1, 2, 3])
[2]
>>> paarisarvude_filter([1, 2, 3, 4, 5])
[2, 4]
>>> paarisarvude_filter([0, -1, 2, -3, 0, 5])
[0, 2, 0]
```

Kui olete juba hulk aega proovinud ülesannet iseseisvalt lahendada ja see ikka ei õnnestu, siis võib-olla saate abi murelahendajalt. Püütud on tüüpilisemaid probleemseid kohti selgitada ja anda vihjeid.

### Nädal 2. Koduülesanne 2.1 Kursused

Tartu Ülikoolis on palju erinevaid õppeaineid ja igal neist on oma ainekood. Näiteks kursuse "Programmeerimise alused II" kood on MTAT.03.256.

Selles ülesandes on teil vaja koostada programm, mis väljastab ainekoodis esineva tähekombinatsiooni alusel vastavate kursuste nimetused ning leitud kursuste koguarvu. Tekstifailis kursused.txt on igal real komaga eraldatult kursuse nimi ja kood.

Koostada programm, mis

- küsib kasutajalt ainekoodi tähekombinatsiooni;
- loeb failist kursuse info kahemõõtmelisse järjendisse;

Abiks võib olla järgnev programmilõik:

```
fail = open("kursused.txt", encoding="UTF-8")
kursused = []
for rida in fail: # iga rea jaoks failist
      osad = rida.strip().split(",") # Jupitame rea koma koha pealt
      kursused.append(osad)
fail.close()
```

- leiab ja väljastab ekraanile õppeainete nimed, mille koodis on nimetatud tähekombinatsioon;
- leiab ja väljastab ekraanile leitud kursuste koguarvu.

Näited programmi tööst

Faili **kursused.txt** sisu:
Andmebaasid,LTAT.O3.OO4
Programmeerimise alused,MTAT.03.236
Tarkvara testimine,LTAT.05.006
Veebilehtede loomine,MTAT.03.297

>>> %Run lahendus.py
  Sisestage ainekoodi neli esimest tähte: MTAT
  Programmeerimise alused
  Veebilehtede loomine
  Leiti 2 sellist kursust!

Faili **kursused.txt** sisu:
Krüptoloogia I,MTAT.07.002
Digitaalne maailmapilt,LTAT.00.020
Mobiilsuse modelleerimine,LTAT.06.016
Infoturve,MTAT.07.028
Transformerid,MTAT.06.055
Võrgutehnoloogia I,LTAT.06.004

>>> %Run 2.1.py
  Sisestage ainekoodi neli esimest tähte: LTAT
  Digitaalne maailmapilt
  Mobiilsuse modelleerimine
  Võrgutehnoloogia I
  Leiti 3 sellist kursust!

Kui olete juba hulk aega proovinud ülesannet iseseisvalt lahendada ja see ikka ei õnnestu, siis võib-olla saate abi murelahendajast.

### Nädal 2. Koduülesanne 2.2 Identsete elementide arv

Sergei soovib luua väikese funktsiooni maatriksist identsete elementide otsimiseks.

Koostada funktsioon **identsete_elementide_arv**, mis

- võtab esimeseks argumendiks täisarvude ja/või sõnede maatriksi ja teiseks argumendiks otsitava elemendi (sõne või täisarv);
- tagastab leitud elementide koguarvu.

Kontrollitakse vaid funktsiooni definitsiooni, programmis seda rakendama ei pea.

Näited funktsiooni tööst

**Näide 1**

```
maatriks1 = [[1, 2, 3, 4, 5],
             [3, 5, 2, 1, 1],
             [4, 2, 3, 1, 2],
             [2, 1, 5, 4, 3],
             [1, 2, 5, 3, 2]]

>>> identsete_elementide_arv(maatriks1,  3)
  5
```

**Näide 2**

```
maatriks2 = [["jah", "ei", "muidugi", "ei", "voib-olla"],
             ["muidugi", "kindlasti", "jah", "jah", "ei"],
             ["kindlasti", "ei", "jah", "voib-olla", "ei"],
             ["kindlasti", "jah", "ei", "jah", "muidugi"],
             ["ei", "jah", "muidugi", "muidugi", "ei"]]

>>> identsete_elementide_arv(maatriks2, "jah")
  7
```

**Näide 3**

```
maatriks3 = [["jah", 5, 1, 3, 2, "voib-olla"],
             [5, 2, "jah", 1, 1, 5],
             ["jah", 3, 3, "ei", 1, 2],
```

```
                ["ei", 1, 3, "voib-olla", "ei", 5],
                [2, "ei", "oh", 5, 2, 1],
                [1, "ei", 2, 2, 1, "jah"]]

>>> identsete_elementide_arv(maatriks3, 5)
  5
```

Vihje.

Elementide loendamisel võib abiks olla meetod **count**.

```
>>> [1, 2, 3, 2, 5].count(2)
  2
```

Kui olete juba hulk aega proovinud ülesannet iseseisvalt lahendada ja see ikka ei õnnestu, siis võib-olla saate abi murelahendajast.

### Nädal 2. Koduülesanne 2.3 Värviliste pallide otsimise võistlus

Hiljuti toimus värviliste pallide otsimise võistlus. Mõte seisnes selles, et väikesesse kohta olid peidetud punased, kollased ja rohelised pallid ning osalejad pidid need üles leidma. Võitjaks sai see, kes kogus kõige rohkem punkte. Punktide arvu mõjutas pallide arv ja värv.

Leitud

- roheline pall andis osalejale 3 punkti;
- kollane pall andis osalejale 2 punkti;
- punane pall andis osalejale 1 punkti.

Mängu lõpus tõid osalejad leitud pallid inspektorile, kes kirjutas pallide värvid tekstifaili. Tekstifailis **tulemused.txt** on iga osaleja kohta rida, mis algab nimega, seejärel on koolon, millele järgnevad pallide värvid, mis on eraldatud komaga.

Koostada programm, mis

- loeb failist võistluse info;
- leiab ja väljastab ekraanile iga osaleja nime ja punktide arvu.

Failist lugeda ja ridu tükeldada saab järgmise programmijupi abil:

```
fail = open("tulemused.txt", encoding="UTF-8")

for rida in fail: # iga rea jaoks failist
      osad = rida.strip().split(":") # Jupitame rea kooloni koha pealt
      pallid = osad[1].split(",")

fail.close()
```

Näited programmi tööst

Faili **tulemused.txt** sisu:
Ants:punane,kollane,roheline,kollane,roheline
Katusha:roheline,punane,roheline,kollane,roheline,punane
Denis:punane,punane,kollane
Misha:roheline,kollane,roheline,kollane

>>>%Run lahendus.py
Ants kogus 11 punkti!
Katusha kogus 13 punkti!
Denis kogus 4 punkti!
Misha kogus 10 punkti!

Kui olete juba hulk aega proovinud ülesannet iseseisvalt lahendada ja see ikka ei õnnestu, siis võib-olla saate abi murelahendajast.

## Nädal 2. Harjutusülesanne 2.4 Maiustused

Hiljuti on avati värvikirev maiustuste pood, mis müüb kolme erinevat tüüpi maiustusi, nagu kooke, marmelaadi ja pulgakomme. Maiustused on erineva värvi ja maitsega, kuid kõik sama tüüpi maiustused on sama hinnaga.

Kauplus salvestab andmed müüdud kaupade arvu kohta tekstifaili **maiustused.txt**, kus iga rida tähistab konkreetset päeva. Andmed on esitatud kolme veeruna, mis on eraldatud tühikuga. Esimene veerg on kookide arv ja viimane veerg on pulgakommide arv.

Koostada programm, mis

- küsib kasutajalt kookide, marmelaadi ja pulgakommide hinda (ujukomaarvud);
- väljastab ekraanile kõigi eri päevadel müüdud kaupade müügitulu eurodes. Tulemus tuleb ümardada kahe kohani peale koma.

Failist kahemõõtmelisse järjendisse saab lugeda järgmise programmijupi abil:

```
fail = open("maiustused.txt", encoding="UTF-8")

müüdud_kaupade_tabel = []

for rida in fail: # iga rea jaoks failist
    kaubad_päevas = [] # kogume ühe päeva kohta info
    osad = rida.split() # tühikute kohalt osadeks

    for osa in osad: # osade kaupa
        kaubad_päevas.append(int(osa)) # järjekordne info juurde

    müüdud_kaupade_tabel.append(kaubad_päevas) # kaubad järjend juurde

fail.close()
```

Näide programmi tööst

Faili **maiustused.txt** sisu:
46 61 14
27 37 48
19 35 28
45 24 49

>>>%Run lahendus.py
Palun sisestage kookide hind: 5.6
Palun sisestage marmelaadi hind: 3.8
Palun sisestage pulgakommide hind: 1.2
Teie müügitulu on 1530.6

Kui olete juba hulk aega proovinud ülesannet iseseisvalt lahendada ja see ikka ei õnnestu, siis võib-olla saate abi murelahendajast.

### Nädal 2. Harjutusülesanne 2.5 Keskmiste väärtuste keskmine väärtus

Aritmeetiline keskmine on lihtsaim viis arvude kogumi keskmise arvutamiseks. Aritmeetilise keskmise leidmiseks liidetakse kõigepealt kõik arvud kokku ja seejärel jagatakse summa arvude arvuga. Siit leiate selle kohta teavet.

On antud kahemõõtmeline järjend ja te peate leidma iga alamjärjendi elementide aritmeetilise keskmise ning seejärel leidma nende alamjärjendi tulemuste aritmeetilise keskmise.

Koostada funktsioon **keskmiste_keskmine**, mis

- võtab argumendiks täisarvude kahemõõtmelise järjendi (mitte tingimata maatriks);
- tagastab keskmiste keskmise ujukomaarvuna, mis on ümardatud kahe kohani peale koma.

Kontrollitakse vaid funktsiooni definitsiooni, programmis seda rakendama ei pea.

Näited funktsiooni tööst

```
>>> keskmiste_keskmine([[5, 2], [6, 5, 4], [9], [1, 5]])
5.12
>>> keskmiste_keskmine([[3, 4, 5], [2, 2, 2, 3], [1], [6, 6]])
3.31
```

Vihje.

Abi võib olla funktsioonide **sum** ja **len** kasutamisest.

Kui olete juba hulk aega proovinud ülesannet iseseisvalt lahendada ja see ikka ei õnnestu, siis võib-olla saate abi murelahendajalt.

### Nädal 2. Harjutusülesanne 2.6 Sümmeetriline maatriks

Sümmeetriline maatriks on maatriks, mille elemendid on samad ülalt alla või vasakult paremale vaadates. Maatriksi numbrid on peegeldatud selle põhidiagonaali suhtes, mis kulgeb vasakust ülanurgast paremasse alanurka. Siit leiate selle kohta teavet.

Sümmeetrilise maatriksi näidet saab näha allpool:

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{pmatrix}$$

Maatriks on sümmeetriline, kui iga element indeksitega i,j on võrdne elemendiga, mille indeksid on j,i. Näites on A[0][1] ja A[1][0] mõlemad võrdsed 2-ga, aga A[1][2] ja A[2][1] võrdsed 5-ga.

Koostada funktsioon **summeetriline_maatriks**, mis

- võtab esimeseks argumendiks täisarvude maatriksi kahemõõtmelise järjendina;
- tagastab boolean-väärtuse True, kui maatriks on sümmeetriline, või False, kui mitte.

Kontrollitakse vaid funktsiooni definitsiooni, programmis seda rakendama ei pea.

Näited funktsiooni tööst

```
>>> summeetriline_maatriks([[1, 2, 3], [2, 4, 5], [3, 5, 6]])
True
>>> summeetriline_maatriks([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
False
>>> summeetriline_maatriks([[2, 4, 6, 8], [4, 9, 10, 12], [6, 10, 20, 25], [8, 12, 25, 30]])
True
```

Kui olete juba hulk aega proovinud ülesannet iseseisvalt lahendada ja see ikka ei õnnestu, siis võib-olla saate abi murelahendajalt.

### Nädal 3. Koduülesanne 3.1 Väikseim ja suurim summa

On antud täisarvulistest väärtustest koosnev kahemõõtmeline järjend ning funktsioon kontrollib iga järjendis olevat järjendit ja tagastab tulemuseks enniku, mis sisaldab kahte elementi. Esimene element ennikus on sisemise järjendi elementide minimaalne summa ja teine element on sisemise järjendi elementide maksimaalne summa.

Koostada funktsioon **vaike_ning_suur**, mis

- võtab argumendiks kahemõõtmelise täisarvude järjendi;
- tagastab enniku, milles on kaks elemendi, millest esimene on sisemise järjendi elementide minimaalne ning teine maksimaalne summa.

Kontrollitakse vaid funktsiooni definitsiooni, programmis seda rakendama ei pea.

Näited funktsiooni tööst

```
>>> vaike_ning_suur([[2, 1, 3], [4, 8], [2, 1, 4]])
  (6, 12)
>>> vaike_ning_suur([[8, 9, 6], [2, 2, 3], [0, 1, 4], [10, 1, 2]])
  (5, 23)
```

Kui olete juba hulk aega proovinud ülesannet iseseisvalt lahendada ja see ikka ei õnnestu, siis võib-olla saate abi murelahendajast.

### Nädal 3. Koduülesanne 3.2 Arvuruutude sõnastik

Koostada funktsioon **arvu_ruudud**, mis

- võtab argumentideks kaks täisarvu;
- tagastab sõnastiku, mille võtmeteks on kõikide arvud alates väiksemast ja väärtusteks vastavate arvude ruudud.

Esimene argument on minimaalne arv, millest astme laiendamine peaks algama, ja teine argument peaks olema maksimaalne arv, mille juures astme laiendamine peaks lõppema. Lihtsuse huvides võib esimest argumenti nimetada `min_arv` ja teist argumenti `max_arv`.

Tagastatav sõnastik peab seega sisaldama kirjeid, kus võtmeteks on arvud, mis algavad esimese argumendiga ja lõpevad teise argumendiga, ning väärtuseks on arvude ruudud (st arvud korrutatud iseendaga).

Pange tähele, et teine argument peab olema suurem kui esimene. Kui esimene argument on suurem kui teine argument, siis tagastatakse tühi sõnastik, näiteks nii {}. Kui esimene ja teine element on sama, siis tagastatakse ühe elemendiga sõnastik.

Kontrollitakse vaid funktsiooni definitsiooni, programmis seda rakendama ei pea.

Näited funktsiooni tööst

```
>>> arvu_ruudud(5, 1)
  {}
>>> arvu_ruudud(1, 1)
  {1: 1}
>>> arvu_ruudud(1, 3)
  {1: 1, 2: 4, 3: 9}
>>> arvu_ruudud(4, 9)
  {4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81}
>>> arvu_ruudud(-4, 0)
  {-4: 16, -3: 9, -2: 4, -1: 1, 0: 0}
```

Kui olete juba hulk aega proovinud ülesannet iseseisvalt lahendada ja see ikka ei õnnestu, siis võib-olla saate abi murelahendajast.

### Nädal 3. Koduülesanne 3.3 Ühine kõigile

Koolivaheajal käis Arthur poodi, et osta endale toidukaupu. Siis otsustas ta huvi pärast uurida, millised tema sõbrad või tuttavad on ostnud samu toidukaupu. Arthur palus neil kirjutada paberile, milliseid toidukaupu nad olid ostnud või anda talle tšekk, kui see neil veel alles on.

Selles ülesandes peate looma funktsiooni, mis loob hulga, mis sisaldab ainult neid elemente (tooted), mis esinevad kõigis antud hulkades.

Koostada funktsioon **uhine_koigile**, mis

- võtab argumendiks hulkade järjendi;
- tagastab hulga, mis sisaldab ainult neid elemente, mis esinevad kõigis järjendis olevates hulkades.

Kontrollitakse vaid funktsiooni definitsiooni, programmis seda rakendama ei pea.

Näited funktsiooni tööst

```
>>> uhine_koigile([{"sidrun", "sai", "piim", "munad"},
                   {"hapukoor", "õunad", "piim"},
                   {"apelsin", "piim", "leib", "jäätis"}])

  {'piim'}

>>> uhine_koigile([{"munad", "tee", "õunad", "kartulikrõpsud", "sai"},
                   {"apelsin", "sai", "sidrun", "munad", "tee", "leib"},
                   {"hapukoor", "leib", "sai", "õunad", "tee", "suhkur", "munad"},
                   {"sai", "tee", "munad", "jäätis", "leib", "apelsin"}])

  {'sai', 'tee', 'munad'}
```

Kui olete juba hulk aega proovinud ülesannet iseseisvalt lahendada ja see ikka ei õnnestu, siis võib-olla saate abi murelahendajast.

## Nädal 3. Harjutusülesanne 3.4 Koordinaadid

Tekstifail **koordinaadid.txt** sisaldab andmeid kahe tähemärgi "x" ja "-" kujul, mis on eraldatud tühikuga. Teie ülesanne on leida x-i asukohad ja kirjutada need kahest elemendist koosneva enniku kujul. Esimene element tähistab rida ja teine element tähistab veergu, kus see asub.

Näiteks (2, 1) näitab, et x asub teises reas ja esimeses veerus.

Koostada funktsioon **koordinaadid**, mis

- võtab argumendiks andmebaasi faili nime;
- tagastab vastava faili sisu põhjal tulemuse järjendina, mis koosneb kahe elemendiga ennikutest.

Kontrollitakse vaid funktsiooni definitsiooni, programmis seda rakendama ei pea.

Näited funktsioonide tööst

Faili **koordinaadid.txt** sisu:
x - - -
- x - x
- - x -
- - - x

>>> koordinaadid("koordinaadid1.txt")
[(1, 1), (2, 2), (2, 4), (3, 3), (4, 4)]

Faili **koordinaadid.txt** sisu:
- - - - - x -
- x x - - - -
- - - - x - -
x - - x - - -

>>> koordinaadid("koordinaadid2.txt")
[(1, 6), (2, 2), (2, 3), (3, 5), (4, 1), (4, 4)]

Kui olete juba hulk aega proovinud ülesannet iseseisvalt lahendada ja see ikka ei õnnestu, siis võib-olla saate abi murelahendajast.

## Nädal 3. Harjutusülesanne 3.5 Lilled ja nende hinnad

Mari töötab kaupluses, kus müüakse erinevaid lilli, ja loob selle jaoks tekstifailide kujul andmebaasi. Mugavuse huvides soovib ta koostada programmi, mille abil saaks näha, millised lilled sobivad teatud hinnaklassi.

Tekstifailid sisaldavad ridu, mis koosnevad lillede nimest ja hinnast eurodes, eraldatuna kooloniga:

Lillad nelgid:2.00
Kollased gerberad:3.50
Roosad liiliad:7.90

1. Koostada funktsioon **failist_sonastik**, mis
    - võtab argumendiks andmebaasi faili nime;
    - tagastab vastava faili sisu põhjal sõnastiku, kus võtmeteks on lillenimed (sõnad) ja väärtusteks hinnad eurodes (ujukomaarv).

2. Koostada funktsioon **sobivad_lilled**, mis
    - võtab argumendiks lillede minimaalse hinna (ujukomaarv), lillede maksimaalse hinna (ujukomaarv) ja eelmise funktsiooni poolt koostatud sõnastiku;
    - tagastab lillenimede hulga, mille hind on võrdne minimaalse või maksimaalse hinnaga või jääb selle vahemiku vahele.

3. Rakendada funktsioone sobivalt programmis, mis
    - küsib kasutajalt
        - andmebaasi faili nime;
        - lillede minimaalset hinda;
        - lillede maksimaalset hinda;
    - väljastab ekraanile hulga lillede nimedega, mis vastavad hindadele;
    - väljastab ekraanile iga sobiva lille nime ja selle hinna eurodes.

Näited funktsioonide tööst

Faili **loikelilled.txt** sisu:
Valged liiliad:7.90
Punased gerberad:3.50
Valged roosid:5.50
Roosad nelgid:2.00
Eukalüpt:9.00
Kollased krüsanteemid:4.50

```
>>> failist_sonastik("loikelilled.txt")
  {'Valged liiliad': 7.9, 'Punased gerberad': 3.5, 'Valged roosid': 5.5, 'Roosad nelgid': 2.0, 'Eukalüpt': 9.0, 'Kollased krüsanteemid': 4.5}

>>> sobivad_lilled(2.5, 5.0, failist_sonastik("loikelilled.txt"))
  {'Punased gerberad', 'Kollased krüsanteemid'}
```

Faili **lillekimbud.txt** sisu:
Peen roosikimp:22.90
Romantiline lillekimp:43.90
Rooside võlu:49.00
Meeldiv lillekimp:33.50
Hingekosutav lillekimp:51.90

```
>>> failist_sonastik("lillekimbud.txt")
   {'Peen roosikimp': 22.9, 'Romantiline lillekimp': 43.9, 'Rooside võlu': 49.0, 'Meeldiv lillekimp': 33.5, 'Hingekosutav lillekimp': 51.9}
```

```
>>> sobivad_lilled(27.5, 49.0, failist_sonastik("lillekimbud.txt"))
  {'Meeldiv lillekimp', 'Romantiline lillekimp', 'Rooside võlu'}
```

Näide programmi tööst

```
Faili loikelilled.txt sisu:
Valged liiliad:7.90
Punased gerberad:3.50
Valged roosid:5.50
Roosad nelgid:2.00
Eukalüpt:9.00
Kollased krüsanteemid:4.50

>>> %Run 3.4.py
  Sisestage andmebaasi faili nimi: loikelilled.txt
  Sisestage minimaalne hind: 3.5
  Sisestage maksimaalne hind: 8.0

  {'Punased gerberad', 'Valged liiliad', 'Kollased krüsanteemid', 'Valged roosid'}
  Punased gerberad maksavad 3.5 eurot (€)
  Valged liiliad maksavad 7.9 eurot (€)
  Kollased krüsanteemid maksavad 4.5 eurot (€)
  Valged roosid maksavad 5.5 eurot (€)
```

Kui olete juba hulk aega proovinud ülesannet iseseisvalt lahendada ja see ikka ei õnnestu, siis võib-olla saate abi murelahendajast.

### Nädal 3. Harjutusülesanne 3.6 Duplikaadid

On antud fail, kus igal real on tühikutega eraldatud täisarvud. Arvude hulk ridadel võib olla erinev. Teie eesmärk on leida igal real korduvate elementide (duplikaatide) arv ja leida rida, kus on neid kõige rohkem. Lugeda kokku kõik korduvate elementide kõik eksemplarid. Näiteks kui ühte elementi on 2 ja teist kolm, siis on koguarv 5.

Koostada programm, mis

- küsib kasutajalt faili nime;
- leiab, milline rida failis sisaldab kõige rohkem duplikaate;
- väljastab ekraanile suurima arvu duplikaatidega rea järjekorranumbri ja duplikaatide arvu.

Ridade järjekorranumbrid algavad tavapäraselt **ühest**.

Näide programmi tööst

Faili **arvud.txt** sisu:
5 12 16 16 7 8  5
1 11 1 12 8 1 11
10 10 15 20 10 15 5
1 1 1 3 1 5 1 3
20 14 25 60 2 8 2

>>> %Run lahendus.py
  Sisestage failinimi: arvud.txt
  Kõige rohkem duplikaate on failis 4. real ja duplikaate on 5.

Vihje.

Ülesande lahendamiseks on kasulik selline andmestruktuur nagu *hulk*.

## Nädal 3. Lisaülesanne 3.7 Ilmatabel failist

Ilmatabel on CSV-failis **ilmaandmed.csv**, kus erinevad päevad on eraldi ridadel. Iga päeva kohta on antud selle kuupäev, minimaalne temperatuur, maksimaalne temperatuur ja õhuniiskus. Eraldajaks on koma.

Temperatuuri mõõdetakse Celsiuse kraadides (°C) ning õhuniiskust protsentides (%), mis näitab veeauru suhtelist sisaldust õhus.

Koostada programm, mis
- loeb failist read;
- väljastab ekraanile keskmise minimaalse temperatuuri;
- väljastab ekraanile keskmise maksimaalse temperatuuri;
- väljastab ekraanile keskmise õhuniiskuse.

Näide programmi tööst

Faili **ilmaandmed.csv** sisu:
2021-01-01,2,5,80
2021-01-02,3,6,75
2021-01-03,1,4,90
2021-01-04,-1,3,85
2021-01-05,0,7,80
2021-01-06,-2,5,70

>>> %Run lahendus.py

```
Keskmine minimaalne temperatuur on 0.5°C
Keskmine maksimaalne temperatuur on 5.0°C
Keskmine õhuniiskus on 80.0%
```

## Nädal 3. Lisaülesanne 3.8 Sõnade lugeja

Teile antakse fail, mis sisaldab teksti.

Koostada programm, mis

- loeb failist teksti;
- väljastab ekraanile sõnastiku, mis sisaldab kirjeid, kus võtmeteks on sõnad ja väärtusteks vastavate sõnade esinemiste arv.

Pange tähele, et sõnastik peaks sisaldama ainult väikese tähega sõnu. Samuti on oluline märkida, et mõne sõna lõpus võib olla punkt või koma. Peate veenduma, et seda sõnastikus ei ole.

Näide programmi tööst

```
Faili tekst.txt sisu:
Mis on programmeerimine
Programmeerimine on lõbus.
Programmeerimine on huvitav.
Programmeerimine on põnev.
Programmeerimine on kõik.

Programmi töö:
>>>%Run 3.7.py
  {'mis': 1, 'on': 5, 'programmeerimine': 5, 'lõbus': 1, 'huvitav': 1, 'põnev': 1, 'kõik': 1}
```

Vihje1.

Väikeste tähtedega sõna tegemiseks võite kasutada meetodit **lower**.

```
>>> "Tulemus".lower()
'tulemus'
```

Vihje2.

Selleks, et eemaldada stringi lõpus olev ebavajalik täht, saate kasutada viilutamist. Näide on näidatud allpool:

```
>>> 'naide.'[:-1]
'naide'
```

## Nädal 4. Koduülesanne 4.1 Kahe tähemärgi tabel

Ülesande eesmärk on koostada tabel, kus iga element sisaldab kahte märki. Funktsioon loob ette antud kahe sõne põhjal kahemõõtmeline järjendi selliselt, et sisemiste järjendite elemendid on kahetähelised sõned, nii sõnede esimesed tähed tulevad esimese sõne põhjal ja teised tähed teise sõne põhjal. Sisemiste järjendite arv sõltub esimese sõne pikkusest ja sisemiste järjendite elementide arv teise sõne pikkusest.

Koostada funktsioon **kahe_tahemargi_tabel**, mis

- võtab argumendiks kaks sõnet;
- ning tagastab kahemõõtmelise järjendi, mis sisaldab elemente kahest sõnest.

Näited funktsiooni eraldi rakendamisest

```
>>> kahe_tahemargi_tabel("abc", "aaf")
[['aa', 'aa', 'af'], ['ba', 'ba', 'bf'], ['ca', 'ca', 'cf']]

>>> kahe_tahemargi_tabel("abcdef", "kk")
[['ak', 'ak'], ['bk', 'bk'], ['ck', 'ck'], ['dk', 'dk'], ['ek', 'ek'], ['fk', 'fk']]
```

Kontrollitakse vaid funktsiooni, programmis seda rakendama ei pea.

Kui olete juba hulk aega proovinud ülesannet iseseisvalt lahendada ja see ikka ei õnnestu, siis võib-olla saate abi murelahendajast.

## Nädal 4. Koduülesanne 4.2 Elektroonika ostmine

Tekstifailis **elektroonika.txt** on andmed eri tüüpi elektroonika müügi kohta kaupluses viimase 7 kuu jooksul. Faili esimeses veerus on toodete nimetused (näiteks nutitelefon, sülearvuti, kõrvaklapid jne) ning järgmises seitsmes veerus igas kuus müüdud ühikute arv. Rea viimane arv on viimase kuu kohta, eelviimane arv eelviimase kuu kohta jne. Andmed on failis eraldatud tühiku abil.

Koostada programm, mis

- loeb failist elektroonika andmed;
- väljastab ekraanile tooded, mille müügi osakaal on tõusnud määratud perioodi algusest lõpuni üle 20%. Algusena võite kasutada esimest kuud ja lõpuna viimast kuud. Protsendid tuleb ümardada täisarvudeks.

Näide programmi tööst

Faili **elektroonika.txt** sisu:
Nutitelefonid 227 229 234 241 259 281 294
Sülearvutid 347 358 372 379 396 415 429
Kõrvaklapid 178 176 181 179 185 191 193
Mikrofonid 162 164 161 168 174 177 181
Tahvelarvutid 257 265 278 289 296 311 321
E-raamatud 72 75 71 78 80 81 84

```
>>> %Run lahendus.py
  Sisestage elektroonikaandmeid sisaldava faili nimi: elektroonika.txt
  Nutitelefonid – müük kasvas 30%.
  Sülearvutid – müük kasvas 24%.
  Tahvelarvutid – müük kasvas 25%.
```

Faili **elektroonika.txt** sisu:
Nutitelefonid 306 302 387 284 286 281 275
Tahvelarvutid 247 255 268 282 291 302 318
Sülearvutid 359 364 373 380 394 414 428
Kõrvaklapid 178 176 170 165 145 146 153
Mikrofonid 112 124 131 144 157 167 182
E-raamatud 70 71 73 68 60 57 58

```
>>> %Run lahendus.py
  Sisestage elektroonikaandmeid sisaldava faili nimi: elektroonika2.txt
  Tahvelarvutid – müük kasvas 29%.
  Mikrofonid – müük kasvas 62%.
```

Kui olete juba hulk aega proovinud ülesannet iseseisvalt lahendada ja see ikka ei õnnestu, siis võib-olla saate abi murelahendajast.

## Nädal 4. Koduülesanne 4.3 Vastupidine tabel

Ülesanne on antud kahemõõtmelises arvude järjendis pöörata kõik järjendis sisalduvad järjendid tagurpidi ja asendada kõik arvud nende negatiivsete väärtustega.

Seda ülesannet tuleb lahendada kahel moel:

1. Koostada funktsioon **vastupidine_tabel**, mis
    - võtab argumendiks kahemõõtmelise arvude järjendi;
    - tagastab uue kahemõõtmelise järjendi, mis on saadud algsest nii, et sisemised järjendid on pööratud tagurpidi ja kõik arvud on korrutatud arvuga -1 (kõigile arvudele on lisatud miinus).

2. Koostada funktsioon **vastupidi_tabel**, mis
    - võtab argumendiks kahemõõtmelise arvude järjendi;
    - pöörab sama järjendi sisemised järjendid tagurpidi ja lisab kõigile elementidele miinuse. Funktsioon ei tagasta midagi.

Mõlemad funktsioonid täidavad sama ülesannet, kuid oluline erinevus seisneb selles, et esimene funktsioon tagastab uue tabeli ilma vana tabelit muutmata, teine aga muudab olemasolevat tabelit ilma midagi tagastamata.

Kontrollitakse vaid funktsioonide definitsioone, programmis neid rakendama ei pea.

Näited funktsiooni tööst

```
>>> tabel = [[2, 4, -1], [0, 5, -3]]
>>> vastupidine_tabel(tabel)
[[1, -4, -2], [3, -5, 0]]
>>> tabel
[[2, 4, -1], [0, 5, -3]]

>>> tabel = [[2, 4, -1], [0, 5, -3]]
>>> vastupidi_tabel(tabel)
>>> tabel
[[1, -4, -2], [3, -5, 0]]
```

Kui olete juba hulk aega proovinud ülesannet iseseisvalt lahendada ja see ikka ei õnnestu, siis võib-olla saate abi murelahendajast.

## Nädal 4. Harjutusülesanne 4.4 Anagraami grupp

Koostada funktsioon **anagrammide_grupeerimine**, mis
- võtab argumendiks erinevate sõnade jarjendi;
- leiab kõik anagrammid ja grupeerib need anagrammide järgi sisemisteks järjenditeks;
- tagastab kahemõõtmelise jarjendi, kus iga sisemine järjend sisaldab sõnade anagramme.

Antud sõne anagramm on sõne, mis on saadud algse sõne tähemärkide ümberpaigutamise teel. Seejuures igat algse sõne tähemärki on uues sõnes kasutatud täpselt üks kord. Lisaks loeme siin, et iga sõne on enda anagramm.

Kontrollitakse vaid funktsiooni definitsiooni, programmis seda rakendama ei pea.

Näited funktsiooni tööst

```
>>> anagrammide_grupeerimine(["vale", "kaubajaam", "laev", "kaubamaja"])
[['vale', 'laev'], ['kaubajaam', 'kaubamaja']]
>>> anagrammide_grupeerimine(["eat", "tea", "tan", "ate", "nat", "bat"])
[['eat', 'tea', 'ate'], ['tan', 'nat'], ['bat']]
```

Vihje1.

Seda ülesannet on mugavam lahendada funktsiooni abil, mis loob stringist sorteeritud jarjendi.

Esimene variant on kasutada funktsiooni **list** ja meetodit sort.

```
>>> naide = list("cba")
>>> naide.sort()
>>> naide
['a', 'b', 'c']
```

Teine variant on kasutada ainult funktsiooni **sorted**.

```
>>> a = sorted("cba")
>>> a
['a', 'b', 'c']
```

Vihje2.

Ülesande lahendamiseks võib osutuda vajalikuks teisendada järjend stringiks. Seda saab teha tsüklis, ühendades elemente. Samuti võite selleks kasutada meetodit **join**.

```
>>> "".join(["a", "b", "c"])
'abc'
```

Vihje3.

Ülesande lahendamiseks võib osutuda vajalikuks leida järjendist indeks teatud elemendi järgi. Seda saab teha tsükli abil, võrreldes elemente. Võite selleks kasutada ka meetodit **index**.

```
>>> ["a", "b", "c"].index("c")
2
```

## Nädal 4. Harjutusülesanne 4.5 Madude liikumine I

Koostada funktsioon **madude_liikumine**, mis

- võtab argumendiks täisarvude maatriksi;
- tagastab järjendi, millesse on kopeeritud maatriksi elemendid järgnevalt: kõigepealt esimene rida järjestuses vasakult paremale, seejärel teine rida järjestuses paremalt vasakule, siis kolmas rida järjestuses vasakult paremale ja nii edasi.

Näited funktsiooni tööst

```
>>> madude_liikumine([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])
[1, 2, 3, 4, 8, 7, 6, 5, 9, 10, 11, 12]
>>> madude_liikumine([[1, 7, 8], [3, 12, 4], [5, 6, 10], [11, 2, 9]])
[1, 7, 8, 4, 12, 3, 5, 6, 10, 9, 2, 11]
```

Kui olete juba hulk aega proovinud ülesannet iseseisvalt lahendada ja see ikka ei õnnestu, siis võib-olla saate abi murelahendajast.

## Nädal 4. Harjutusülesanne 4.6 Maagiline ruut

Teie eesmärk selles ülesandes on koostada funktsioon **on_maagiline_ruut** koos teiste funktsioonidega, mis kontrollib, kas antud ruutmaatriks on maagiline ruut.

Maagiline ruut ehk võluruut on ruudukujuline arvude tabel, mille igas reas, igas veerus ja mõlemal diagonaalil olevate naturaalarvude summa on võrdne. Maagiliste ruutude kohta saad lugeda siit eesti keeles ja siit inglise keeles.

Kui näiteks maatriksi esimese rea elementide summa on 15, siis teistes ridades, veergudes ja diagonaalidel peab olema täpselt sama summa.

1. Koostada funktsioon **on_maagiline_ruut**, mis
    - võtab argumendiks täisarvude ruutmaatriksi;
    - tagastab tõeväärtuse, mis näitab, kas ruut on maagiline või mitte. Selle sammu lõpuleviimiseks peate tegema järgmist.

2. Koostada funktsioon **vaatame_read**, mis
    - võtab argumendiks maatriksi ja summa, mis peab olema kõigil ridadel;
    - tagastab tõeväärtuse, mis sõltub sellest, kas kõik ruudu ridade elementide summad vastavad soovitud summale või mitte. Funktsioon tuleks välja kutsuda funktsioonis **on_maagiline_ruut** ja seejärel selle tulemust seal kontrollida.

3. Koostada funktsioon **vaatame_veerud**, mis
    - võtab argumendiks maatriksi ja summa, mis peab olema kõikides veergudes;
    - tagastab tõeväärtuse, mis sõltub sellest, kas kõik ruudu veergude elementide summad vastavad nõutavale summale või mitte. Funktsioon tuleb välja kutsuda funktsioonis **on_maagiline_ruut** ja seejärel selle tulemust seal kontrollida.

4. Koostada funktsioon **vaatame_peadiagonaali**, mis
    - võtab argumendiks maatriksi ja summa, mis peaks olema peadiagonaalil;
    - tagastab tõeväärtuse, mis sõltub sellest, kas ruudu peadiagonaali elementide summa on võrdne nõutava summaga või mitte. Funktsioon tuleb välja kutsuda funktsioonis **on_maagiline_ruut** ja seejärel selle tulemust seal kontrollida.

5. Koostada funktsioon **vaatame_korvaldiagonaali**, mis
    - võtab argumendiks maatriksi ja summa, mis peaks olema kõrvaldiagonaalil;
    - tagastab tõeväärtuse, mis sõltub sellest, kas ruudu kõrvaldiagonaali elementide summa on võrdne nõutava summaga või mitte. Funktsioon tuleb välja kutsuda funktsioonis **on_maagiline_ruut** ja seejärel selle tulemust seal kontrollida.

Näited funktsiooni tööst

```
>>> vaatame_read([[4, 9, 2], [3, 5, 7], [8, 1, 6]], 15)
True
>>> vaatame_veerud([[4, 9, 2], [3, 5, 7], [8, 1, 6]], 15)
True
>>> vaatame_peadiagonaali([[4, 9, 2], [3, 5, 7], [8, 1, 6]], 15)
True
>>> vaatame_korvaldiagonaali([[4, 9, 2], [3, 5, 7], [8, 1, 6]], 15)
True
>>> on_maagiline_ruut([[4, 9, 2], [3, 5, 7], [8, 1, 6]])
True
>>> on_maagiline_ruut([[1, 4, 5], [7, 8, 2], [3, 9, 6]])
False
```

## Nädal 4. Lisaülesanne 4.7 Madude liikumine II

Koostada funktsioon **madude_liikumine_II**, mis

- võtab argumendiks täisarvude maatriksi;
- tagastab järjendi, millesse on kopeeritud maatriksi elemendid järgnevalt: kõigepealt esimene veerg järjestuses ülevalt alla, seejärel teine veerg järjestuses alt üles, siis kolmas veerg järjestuses ülevalt alla jne.

Näited funktsiooni tööst

```
>>> madude_liikumine_II([[12, 11, 10, 9], [8, 7, 6, 5], [1, 2, 3, 4]])
[12, 8, 1, 2, 7, 11, 10, 6, 3, 4, 5, 9]
>>> madude_liikumine_II([[1, 7, 8], [3, 12, 4], [5, 6, 10], [11, 2, 9]])
[1, 3, 5, 11, 2, 6, 12, 7, 8, 4, 10, 9]
```

Kui olete juba hulk aega proovinud ülseannet iseseisvalt lahendada ja see ikka ei õnnestu, siis võib-olla saate abi murelahendajast.

## Nädal 5. Koduülesanne 5.1 Üles ja alla

Koostada rekursiivne funktsioon **ules_alla**, mis võtab argumendiks positiivse täisarvu.

1. Kui argument on **paaris**, siis väljastab funktsioon ekraanile kahanevalt positiivsed paaritud arvud alates arvust, mis ei ole suurem antud argumendist, seejärel sõne Põhi! ning siis kasvavalt paarisarvud alates 2-st kuni viimase paarisarvuni, mis on argument.

2. Kui argument on **paaritu**, siis väljastab funktsioon ekraanile kahanevalt positiivsed paarisarvud alates arvust, mis ei ole suurem antud argumendist, kuni 0, seejärel sõne Põhi! ning siis kasvavalt paaritud arvud alates 1-st kuni viimase paaritu arvuni, mis on argument.

Kontrollitakse vaid funktsiooni definitsiooni, programmis seda rakendama ei pea.

Näited funktsiooni eraldi rakendamisest

```
>>> ules_alla(0)
Põhi!
>>> ules_alla(1)
0
Põhi!
1
>>> ules_alla(3)
```

```
2
0
Põhi!
1
3
>>> ules_alla(6)
5
3
1
Põhi!
2
4
6
```

Vihje

Rekursiivses funktsioonis täidetakse "alla minnes" laused, mis on enne rekursiivset väljakutset ja "üles tulles" laused, mis on pärast rekursiivset väljakutset.

Kui olete juba hulk aega proovinud ülesannet iseseisvalt lahendada ja see ikka ei õnnestu, siis võib-olla saate abi murelahendajast.

### Nädal 5. Koduülesanne 5.2 Paaritu summa

Koostada rekursiivne funktsioon **paaritu_summa**, mis

- võtab argumendiks ühe positiivse täisarvu;
- tagastab kõikide positiivsete paaritute arvude summa, mis on väiksemad või võrdsed argumendiga.

Kontrollitakse vaid funktsiooni definitsiooni, programmis seda rakendama ei pea.

Näited funktsiooni eraldi rakendamisest

```
>>> paaritu_summa(1)
1
>>> paaritu_summa(2)
1
>>> paaritu_summa(3)
4
>>> paaritu_summa(100)
2500
>>> paaritu_summa(101)
```

```
2601
```

Vihje

Abiks võib olla tähelepanek, et paarisarvulise argumendi puhul on paaritu summa sama, mis eelneva paaritu arvu puhul: paaritu_summa(8) on sama, mis paaritu_summa(7). Ühes rekursiivses funktsioonis võib olla ka mitu rekursiivset väljakutset erinevate argumentidega.

Oma funktsiooni saate lihtsalt testida, kirjutades sama funktsionaalsusega mitterekursiivse funktsiooni ning kontrollides, kas need funktsioonid annavad sama tulemuse.

Kui olete juba hulk aega proovinud ülesannet iseseisvalt lahendada ja see ikka ei õnnestu, siis võib-olla saate abi murelahendajast.

### Nädal 5. Harjutusülesanne 5.3 Tärnid

Koostada rekursiivne funktsioon **tarnide_fraktaal**, mis

- võtab argumendiks ühe positiivse täisarvu n;
- väljastab ekraanile püramiidilaadse tähtede mustri. Püramiidi keskel oleva stringi pikkus võrdub arvuga n ja äärtes on see võrdne 1-ga. Iga rekursiooni tase lisab püramiidile erineva rea, luues struktuuri. Funktsioon ei peaks midagi tagastama.

Fraktaal koosneb tähtedest - "🌟". Saate selle sealt kopeerida ja kleepida funktsiooni sisse stringina.

Kontrollitakse vaid funktsiooni definitsiooni, programmis seda rakendama ei pea

Näited funktsiooni tööst

```
>>> tarnide_fraktaal(0)
>>> tarnide_fraktaal(1)
🌟
>>> tarnide_fraktaal(2)
🌟
🌟🌟
🌟
>>> tarnide_fraktaal(3)
🌟
🌟🌟
🌟
🌟🌟🌟
🌟
```

Vihje.

Selleks, et teha soovitud pikkusega string konkreetse tähemärgiga, saate teha järgmise toimingu:

```
>>> '🌟' * 2
'🌟🌟'

>>> '🌟' * 5
'🌟🌟🌟🌟🌟'
```

## Nädal 5. Harjutusülesanne 5.4 Arvutame lõpuni

Koostada rekursiivne funktsioon **arvutame_lopuni**, mis

- võtab argumentideks kaks positiivset täisarvu n ja m;
- tagastab saadud summa;
- väljastab kõik liidetavad ekraanile.

Funktsioon peab rekursiivselt korrutama praeguse arvu kordajaga, mis väheneb iga rekursiooni sammuga ühe võrra. Igal rekursioonisammul suurendatakse praegust arvu korrutamise tulemuse võrra ja funktsioon tagastab kõigi igal sammul saadud arvude kogusumma. Funktsioon peab kõik liidetavad ekraanile väljastama.

Kui arv on 2 ja kordaja on 5, siis saadakse järgmine liitmine: 2 + (2 * 5) + (10 * 4) + (40 * 3) + (120 * 2) + (240 * 1) = 412

Kontrollitakse vaid funktsiooni definitsiooni, programmis seda rakendama ei pea.

Näited funktsiooni tööst

```
>>> arvutame_lopuni(1, 3)
1
3
6
10
>>> arvutame_lopuni(1, 5)
1
5
20
```

```
60
120
206
>>> arvutame_lopuni(2, 4)
2
8
24
48
82
>>> arvutame_lopuni(3, 5)
3
15
60
180
360
618
```

Kui olete juba hulk aega proovinud ülesannet iseseisvalt lahendada ja see ikka ei õnnestu, siis võib-olla saate abi murelahendajast.

### Nädal 5. Harjutusülesanne 5.5 Algarvude summeerimine (Väike kordamine)

**Algarvud** on need arvud, mis on suuremad kui 1 ning jaguvad ainult iseenda ja ühega. Kui arv jagub lisaks ka mõne muu positiivse täisarvuga, siis pole tegemist algarvuga. Lisateavet leiate siit.

Järgmine programmilõik väljastab arvude järjendi puhul nende arvude arvu, mis on algarvud:

```
arvude_jarjend = [4, 1, 11, 6, 7, 22]

algarvude_sum = 0
for arv in arvude_jarjend:
    algarv = True
    if arv <= 1:
        algarv = False
    else:
        for alam_arv in range(2, arv):
            if arv % alam_arv == 0:
                algarv = False
                break
    if algarv:
        algarvude_sum += 1

print("Algarvud on kokku " + str(algarvude_sum))
```

Teie ülesanne on koostada funktsioon **on_algarv** nii, et alltoodud programmilõik töötaks ülaltooduga võrdväärselt mistahes järjendi puhul:

```
arvude_jarjend = [4, 1, 11, 6, 7, 22]

algarvude_sum = 0
for arv in arvude_jarjend:
    algarvude_sum += on_algarv(arv)

print("Algarvud on kokku " + str(algarvude_sum))
```

Toodud programmilõigule tuleb lisada funktsiooni on_algarv definitsioon. Antud koodijupist on näha, kuidas seda rakendatakse. Etteantud koodi muuta pole vaja. Kui korrektne definitsioon on lisatud, siis peaksid mõlemad koodilõigud andma täpselt sama tulemuse (iga kahemõõtmelise arvude järjendi puhul).

Kontrollitakse vaid funktsiooni on_alglopuline definitsiooni, selle rakendamist näitama ei pea.

Lahendamisel on abiks Kordamine materjal.

## Nädal 5. Harjutusülesanne 5.6 Alg ja lõpp

Järgmine mitterekursiivne funktsioon **alg_ja_lopp** arvutab järjendi esimese ja viimase elemendi korrutiste summa, vähendades järjestikku järjendit, jättes välja töödeldud elemendid, kuni järjendisse jääb üks või null elementi. Kui lõppu jääb üks element, siis liidetakse tulemusele selle elemendi ruut.

```
def alg_ja_lopp(jarjend):
    tulemus = 0
    while(len(jarjend) > 1):
        korrutis = jarjend[0] * jarjend[-1]
        tulemus += korrutis
        jarjend = jarjend[1:-1]
    if len(jarjend) == 1:
        tulemus += jarjend[0] * jarjend[0]
    return tulemus
```

Koostada rekursiivne funktsioon **alg_ja_lopp_rek** nii, et see funktsioon töötaks samamoodi nagu ülaltoodud mitterekursiivne funktsioon. Rekursiivne ja mitterekursiivne funktsioon võtavad argumendiks arvude järjendi ja tagastavad täisarvu, mis on korrutiste summa.

Kui funktsioon alg_ja_lopp_rek on korrektselt tehtud, siis peaksid mõlemad funktsioonid andma täpselt sama tulemuse (iga arvude järjendi puhul).

Kontrollitakse vaid funktsiooni alg_ja_lopp_rek definitsiooni, selle rakendamist näitama ei pea.

## Nädal 5. Lisaülesanne 5.7 Elementide paaride vahetamine

Koostada rekursiivne funktsioon **paaride_vahetamine**, mis

- võtab argumendiks arvude järjendi;
- tagastab uue järjendi, milles elemendid vahetavad kohad igas naaberpaaris. Kui järjendil on paaritu arv elemente, siis jääb viimase elemendi väärtus muutumatuks.

Kontrollitakse vaid funktsiooni definitsiooni, programmis seda rakendama ei pea.

Näited funktsiooni tööst

```
>>> paaride_vahetamine([1, 2])
[2, 1]
>>> paaride_vahetamine([1, 2, 5])
[2, 1, 5]
>>> paaride_vahetamine([-1, 1, 2, -3, 5])
[1, -1, -3, 2, 5]
```

## Nädal 5. Lisaülesanne 5.8 Pikim sügavus

Koostada funktsioon **pikim_sugavus**, mis

- võtab argumendiks suvalise sügavusega järjendi;
- tagastab arvu, mis näitab kõige suuremat sügavust.

Sisendjärjendi elemendid võivad olla järjendid, mille elemendid võivad omakorda olla järjendid jne. Siiski võivad need sisaldada ka muid elemente, mis ei ole järjendid.

Järjendi sügavus on defineeritud kui algses järjestuses pesastunud järjendi tasemete arv. Teisisõnu, sügavus mõõdab, kui sügaval asuvad elemendid pesastatud struktuurides.

Kontrollitakse vaid funktsiooni definitsiooni, programmis seda rakendama ei pea.

Näited funktsiooni tööst

```
>>> pikim_sugavus([3, 2, 1])
1
>>> pikim_sugavus([[[3, 5, 7]]])
3
>>> pikim_sugavus([[[], 3], [2, 3]])
3
```

```
>>> pikim_sugavus([[[], [1, [[5]]]], [2, 3]])
5
>>> pikim_sugavus([[[[[[[[]]]]]]]])
7
>>> pikim_sugavus(5)
0
```

### Nädal 6. Koduülesanne 6.1 Kehamassiindeks

Loo klass **Inimene**, millel on kolm isendivälja:

- nimi,
- pikkus (meetrites),
- kehakaal (kilogrammides).

Lisa klassile konstruktor, mis omistab väljadele etteantud väärtused. Pange tähele, et kehakaal ja pikkus on ujukomaarvud.

Lisa klassile Inimene meetod **arvuta_indeks**, mis tagastab inimese kehamassiindeksi. Kehamassiindeksi (KMI) arvutamiseks saate leida kalkulaatorid siit ja siit. Kehamassiindeksi arvutamise valem on esitatud allpool:

$$\text{KMI} = \text{kehakaal (kg)} / \text{pikkus (m)}^2$$

Koostada põhiprogramm, mis

- küsib kasutajalt esimese inimese nime, pikkust ja kehakaalu;
- küsib kasutajalt teise inimese nime, pikkust ja kehakaalu;
- loob kaks klassi Inimene isendit;
- väljastab ekraanile mõlema inimese kohta nime ja kehamassiindeksi.

Näide programmi tööst

```
>>> %Run lahendus.py
Sisesta esimese inimesi nimi: Aleksander
Sisesta esimese inimesi pikkus (m): 1.82
Sisesta esimese inimesi kehakaal (kg): 89
Sisesta teise inimesi nimi: Raivo
Sisesta teise inimesi pikkus (m): 1.78
Sisesta teise inimesi kehakaal (kg): 80.8
Aleksander, kehamassiindeks on 26.87.
Raivo, kehamassiindeks on 25.5.
```

Kui olete juba hulk aega proovinud ülesannet iseseisvalt lahendada ja see ikka ei õnnestu, siis võib-olla saate abi murelahendajast.

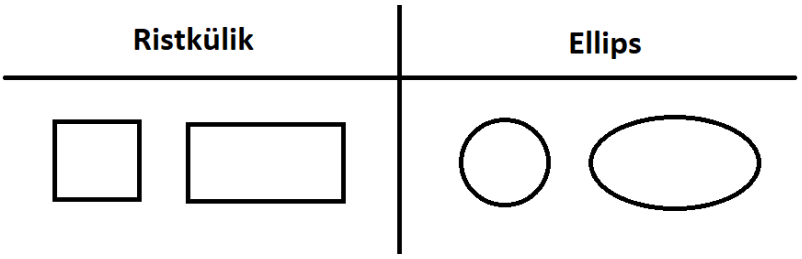
## Nädal 6. Koduülesanne 6.2 Tordid

Loo klass **Tort**, milles on isendiväljad

- nimi,
- kuju (põhiprogrammis väärtuseks kas "ristkulik" või "ellips"),
- pikkus (sentimeetrites),
- laius (sentimeetrites),
- kõrgus (sentimeetrites).

Lisa klassile konstruktor, mis omistab kõigile isendiväljadele etteantud väärtused.

Samuti lisa klassile meetod **ruumala**, mis tagastab tordi ruumala ümardatuna kümnendikeni.




Tordi ruumala sõltub selle kujust, pikkusest, laiusest ja kõrgusest. Ruumala leidmiseks saab kasutada järgmisi valemeid:

- Ristküliku ruumala = pikkus x laius x kõrgus
- Ellipsi ruumala = $\pi$ x $\frac{pikkus}{2}$ x $\frac{laius}{2}$ x kõrgus

Tordi andmed on failis nii, et iga tordi nimi, kuju, pikkus, laius ja kõrgus on ühel real eraldatud tühikuga (vt näide). Failis on kahe tordi andmed (failis ei ole rohkem ridu kui kaks).

Koostada põhiprogramm, mis

- loeb failist **tordid.txt** kahe isiku andmed;
- loob kaks Tort klassi isendit;
- väljastab ekraanile tortide ruumala;
- teeb kindlaks, kas esimene tort on suurem või väiksem kui teine, ja väljastab teate <nimi> on <suurem/väiksem> kui <nimi>.

Näited programmi tööst

```
Faili tortid.txt sisu:
Lirisetort ellips 26.4 26.4 7
Biskviit ristkulik 26.4 26.4 7

>>> %Run lahendus.py
  Lirisetort - pindala on 3831.7 cm3
  Biskviit - pindala on 4878.7 cm3
  Lirisetort on väiksem kui Biskviit

Faili tortid.txt sisu:
Lirisetort ellips 20 14 5.8
Biskviit ristkulik 16 12.5 4.5

>>> %Run lahendus.py
  Lirisetort - pindala on 1275.5 cm3
  Biskviit - pindala on 900.0 cm3
  Lirisetort on suurem kui Biskviit
```

Vihje.

π (hääldatakse "pi") on matemaatiline konstant, mis võrdub ringi ümbermõõdu ja selle läbimõõdu suhtega. Selle kasutamiseks programmis saab kasutada moodulit **math** ja seejärel selle mooduli **pi** meetodit:

```
>>> import math
>>> math.pi
3.141592653589793
```

Kui olete juba hulk aega proovinud ülesannet iseseisvalt lahendada ja see ikka ei õnnestu, siis võib-olla saate abi murelahendajast.

## Nädal 6. Harjutusülesanne 6.3 Õpilane ning hinded

Loo klass **Opilane**, milles on isendiväljad

- nimi,
- vanus,
- hinded (järjend. kus on täisarvud vahemikus 0 kuni 5).

Lisa klassile konstruktor, mis omistab kõigile isendiväljadele etteantud väärtused.

Selle klassi jaoks tuleb koostada kolm meetodit:

1. Lisa klassile meetod **keskmine_hinne**, mis tagastab keskmise hinde. Keskmise hinde saamiseks liidame kõik loetelus olevad arvud kokku ja jagame nende arvuga.

2. Lisa klassile meetod **lisame_hinde**, mis võtab argumendiks hinde ja lisab selle hinde järjendisse. Meetod ei pea midagi tagastama.

3. Lisa klassile meetod **prindime_info**, mis väljastab ekraanile teavet, mis sisaldab õpilase nime ja vanust. Meetod ei pea midagi tagastama.

Kontrollitakse vaid klassi ja selle meetodite definitsiooni, programmis seda rakendama ei pea.

Näide klassi tööst

```
>>> meie_opilane = Opilane("Roman", 17, [5, 4, 3, 5])
>>> meie_opilane.keskmine_hinne()
4.2
>>> meie_opilane.lisame_hinde(5)
>>> meie_opilane.keskmine_hinne()
4.4
>>> meie_opilane.prindime_info()
Tudeng: nimi = Roman, vanus = 17.
```

## Nädal 6. Harjutusülesanne 6.4 Andmete uuendamine (harjutamine kontrolltööks)

Koostada funktsioon **uuendatud_andmed**, mis

- võtab argumentideks kahemõõtmelise järjendi, mis sisaldab algseid andmeid, ja sõnastiku;
- tagastab uuendatud kahemõõtmelise järjendi.

Funktsioon **uuendatud_andmed** uuendab sõnastiku põhjal teatavaid väärtusi kahemõõtmelises järjendis. Sõnastiku võtmed näitavad, millised väärtused tuleb asendada kahemõõtmelises järjendis. Vastavad sõnastiku väärtused näitavad, millega need elemendid tuleks asendada.

Samuti on oluline, et kahemõõtmeline järjend ei muutuks funktsioonis. See tähendab, et pärast funktsiooni väljakutsumist peab kahemõõtmeline järjend olema täpselt sama, mis ta oli enne funktsiooni väljakutsumist.

Kontrollitakse vaid funktsiooni definitsiooni, programmis seda rakendama ei pea.

Näited funktsiooni tööst

```
>>> andmed = [[2, 1], [4, 3]]
>>> sonastik = {1: 5, 3: 10}
>>> uuendatud_andmed(andmed, sonastik)
[[2, 5], [4, 10]]
>>> andmed
[[2, 1], [4, 3]]

>>> andmed = [[1, 4, 5, 6], [2, 3, 5, 6], [2, 6, 1, 1]]
>>> sonastik = {1: 5, 2: 7, 5: 1}
>>> uuendatud_andmed(andmed, sonastik)
[[5, 4, 1, 6], [7, 3, 1, 6], [7, 6, 5, 5]]
>>> andmed
[[1, 4, 5, 6], [2, 3, 5, 6], [2, 6, 1, 1]]

>>> andmed = [[2, 3, 2], [5, 7, 5], [3, 5, 3], [7, 2, 7]]
>>> sonastik = {2: 4, 3: 9}
>>> uuendatud_andmed(andmed, sonastik)
[[4, 9, 4], [5, 7, 5], [9, 5, 9], [7, 4, 7]]
>>> andmed
[[2, 3, 2], [5, 7, 5], [3, 5, 3], [7, 2, 7]]
```

## Nädal 6. Harjutusülesanne 6.5 Projektide finantstulemuste analüüsimine (harjutamine kontrolltööks)

Yuku kirjutas failis erinevate projektide nimed, mis on ettevõtted, ning nende tulud ja kulud. Ta soovis luua programmi, mis paluks sisestada faili nime ja projekti nime, et seejärel näidata selle kasumit ning samuti teiste projektide arvu, mille kasumid ei ole nii erinevad.

Failis **projektid.txt** on projektide nimetused, nende tulud ja kulud. Andmed on eraldatud tühikuga.

Faili **projektid.txt** sisu:
Autodikud 3100 1270
Stardi_Pood 2480 460
Sada_ja_uks 2690 550
Patersoni_Pood 3610 1400
Sarkras 2020 750
Vanaisa_Pood 2225 780

Koostada funktsioon **projektid_sõnastikku**, mis

- võtab argumendiks faili nime;
- tagastab sõnastiku, mille võtmeteks on projektide nimed ja väärtusteks projektide kasum.

Kasum on ettevõtte (projekti) tulude ja kulude vahe, mis näitab, kui palju ettevõte on teeninud. Kasumi saamiseks tuleb tuludest maha arvata kulud.

Näide funktsiooni tööst

```
>>> projektid_sõnastikku("projektid.txt")
{'Autodikud': 1830, 'Stardi_Pood': 2020, 'Sada_ja_uks': 2140, 'Patersoni_Pood': 2210, 'Sarkras': 1270, 'Vanaisa_Pood': 1445}
```

Koosta põhiprogramm, mis

- küsib kasutajalt
    - faili nime;
    - selle projekti nime, mille kohta ta soovib saada teavet;
- kasutades funktsiooni projektid_sõnastikku abil loodud sõnastikku, väljastab ekraanile projekti nime ja selle kasumi kujul <projekti_nimi> - tulude ja kulude vahe on <projekti_kasum> eurot.,
    - kui kasutaja sisestab projekti nime, mida sõnastikus ei ole, väljastab programm teate "Sellist projekti ei ole sõnastikus";
- leiab ja väljastab ekraanile teiste projektide arvu, mille kasum erineb valitud projekti kasumist mitte rohkem kui 200 euro võrra.

Näited programmi tööst

```
>>> %Run Lahendus.py
Sisesta faili nimi: projektid.txt
Sisesta projekti nimi: Sada_ja_uks
Sada_ja_uks - tulude ja kulude vahe on 2140 eurot.
On leidnud 2 peaaegu sama tulemusena projekti.

>>> %Run Lahendus.py
Sisesta faili nimi: projektid.txt
Sisesta projekti nimi: Vanaisa_Pood
Vanaisa_Pood - tulude ja kulude vahe on 1445 eurot.
On leidnud 1 peaaegu sama tulemusena projekti.

>>> %Run Lahendus.py
Sisesta faili nimi: projektid.txt
```

Sisesta projekti nimi: Vana_Pood
Sellist projekti ei ole sõnastikus.

Vihje.

Kontrollimaks, et üks väärtus ei ole teisest suurem ega väiksem kui 100 eurot, saab kasutada järgmist valemit:

abs(väärtus1 - väärtus2) <= 100

## II Koostatud murelahendajad

Nädal 1. Ülesanne 1. : https://progtugi.cs.ut.ee/#/ts/65d0b0ac7f849fea14e8b28d/

Nädal 1. Ülesanne 2. : https://progtugi.cs.ut.ee/#/ts/65d0dd727f849fea14e8b43a/

Nädal 1. Ülesanne 3. : https://progtugi.cs.ut.ee/#/ts/65d10ab17f849fea14e8b4b4/

Nädal 1. Ülesanne 4. : https://progtugi.cs.ut.ee/#/ts/65d1f8467f849fea14e8b711/

Nädal 1. Ülesanne 5. : https://progtugi.cs.ut.ee/#/ts/65d20e857f849fea14e8b800/

Nädal 2. Ülesanne 1. : https://progtugi.cs.ut.ee#/ts/65db06737f849fea14e8f748/

Nädal 2. Ülesanne 2. : https://progtugi.cs.ut.ee/#/ts/65d2474b7f849fea14e8b9de/

Nädal 2. Ülesanne 3. : https://progtugi.cs.ut.ee/#/ts/65db16017f849fea14e8f82e/

Nädal 2. Ülesanne 4. : https://progtugi.cs.ut.ee/#/ts/65ddfc487f849fea14e917db/

Nädal 2. Ülesanne 5. : https://progtugi.cs.ut.ee/#/ts/65d34a537f849fea14e8bd77/

Nädal 2. Ülesanne 6. : https://progtugi.cs.ut.ee/#/ts/65de0c4b7f849fea14e918e8/

Nädal 3. Ülesanne 1. : https://progtugi.cs.ut.ee/#/ts/65d36d087f849fea14e8bee7/

Nädal 3. Ülesanne 2. : https://progtugi.cs.ut.ee/#/ts/65d3843e7f849fea14e8c0ac/

Nädal 3. Ülesanne 3. : https://progtugi.cs.ut.ee/#/ts/65db22927f849fea14e8f9a9/

Nädal 3. Ülesanne 4. : https://progtugi.cs.ut.ee/#/ts/65de11967f849fea14e9195e/

Nädal 3. Ülesanne 5. : https://progtugi.cs.ut.ee/#/ts/65feaa957f849fea14e9f9ae/

Nädal 4. Ülesanne 1. : https://progtugi.cs.ut.ee/#/ts/65db30687f849fea14e8fa81/

Nädal 4. Ülesanne 2. : https://progtugi.cs.ut.ee/#/ts/65db4bd77f849fea14e8fd1d/

Nädal 4. Ülesanne 3. : https://progtugi.cs.ut.ee/#/ts/65d485cd7f849fea14e8c461/

Nädal 4. Ülesanne 5. : https://progtugi.cs.ut.ee/#/ts/65d4b1ef7f849fea14e8c8ff/

Nädal 4. Ülesanne 7. : https://progtugi.cs.ut.ee/#/ts/65fac0b97f849fea14e9e8cd/

Nädal 5. Ülesanne 1. : https://progtugi.cs.ut.ee/#/ts/65db54527f849fea14e8fe2d/

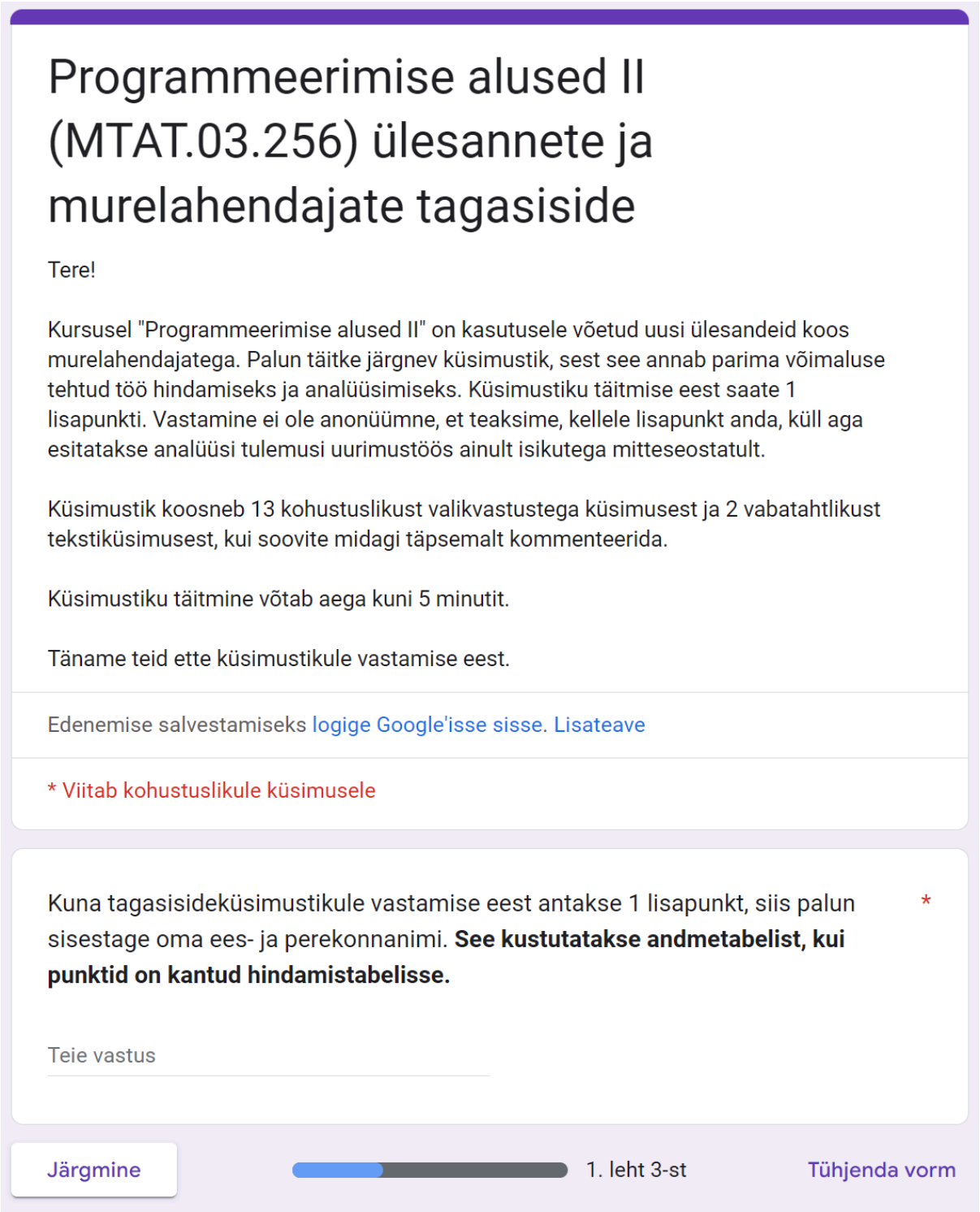
Nädal 5. Ülesanne 2. : https://progtugi.cs.ut.ee/#/ts/65db6a8b7f849fea14e900fa/

Nädal 5. Ülesanne 4. : https://progtugi.cs.ut.ee/#/ts/65fec8ad7f849fea14e9fa37/

Nädal 6. Ülesanne 1. : https://progtugi.cs.ut.ee/#/ts/65db73ac7f849fea14e901fc/

Nädal 6. Ülesanne 2. : https://progtugi.cs.ut.ee/#/ts/65db8d167f849fea14e904a1/

## III Koostatud tagasiside küsimustik



# Programmeerimise alused II (MTAT.03.256) ülesannete ja murelahendajate tagasiside

Tere!

Kursusel "Programmeerimise alused II" on kasutusele võetud uusi ülesandeid koos murelahendajatega. Palun täitke järgnev küsimustik, sest see annab parima võimaluse tehtud töö hindamiseks ja analüüsimiseks. Küsimustiku täitmise eest saate 1 lisapunkti. Vastamine ei ole anonüümne, et teaksime, kellele lisapunkt anda, küll aga esitatakse analüüsi tulemusi uurimustöös ainult isikutega mitteseostatult.

Küsimustik koosneb 13 kohustuslikust valikvastustega küsimusest ja 2 vabatahtlikust tekstiküsimusest, kui soovite midagi täpsemalt kommenteerida.

Küsimustiku täitmine võtab aega kuni 5 minutit.

Täname teid ette küsimustikule vastamise eest.

Edenemise salvestamiseks logige Google'isse sisse. Lisateave

* Viitab kohustuslikule küsimusele

Kuna tagasisideküsimustikule vastamise eest antakse 1 lisapunkt, siis palun sisestage oma ees- ja perekonnanimi. **See kustutatakse andmetabelist, kui punktid on kantud hindamistabelisse.** *

Teie vastus

Järgmine 1. leht 3-st Tühjenda vorm

## Nädal 1

Järgnevad küsimused puudutavad esimese nädala praktikumi ülesandeid koos murelahendajatega.

Ülesanded:
1.1 Temperatuuri teisendamine
1.2 Raha kogumine
1.3 Elekter
1.4 Elektriseadmed
1.5 Paarisarvude filtreerimine

Palun hinnake alljärgneva skaala alusel, kas 1. nädala praktikumi ülesanded keskenduvad põhiteemadele. *

| | 1 - ei nõustu | 2 | 3 | 4 | 5 - nõustun | Ei lahendanud seda ülesannet |
|---|---|---|---|---|---|---|
| 1.1 Temperatuuri teisendamine | ○ | ○ | ○ | ○ | ○ | ○ |
| 1.2 Raha kogumine | ○ | ○ | ○ | ○ | ○ | ○ |
| 1.3 Elekter | ○ | ○ | ○ | ○ | ○ | ○ |
| 1.4 Elektriseadmed | ○ | ○ | ○ | ○ | ○ | ○ |
| 1.5 Paarisarvude filtreerimine | ○ | ○ | ○ | ○ | ○ | ○ |

Palun hinnake alljärgneva skaala alusel, kas 1. nädala praktikumi ülesanded on selgelt sõnastatud. *

| | 1 - ei nõustu | 2 | 3 | 4 | 5 - nõustun | Ei lahendanud seda ülesannet |
|---|---|---|---|---|---|---|
| 1.1 Temperatuuri teisendamine | ○ | ○ | ○ | ○ | ○ | ○ |
| 1.2 Raha kogumine | ○ | ○ | ○ | ○ | ○ | ○ |
| 1.3 Elekter | ○ | ○ | ○ | ○ | ○ | ○ |
| 1.4 Elektriseadmed | ○ | ○ | ○ | ○ | ○ | ○ |
| 1.5 Paarisarvude filtreerimine | ○ | ○ | ○ | ○ | ○ | ○ |

Palun hinnake alljärgneva skaala alusel, kas 1. nädala praktikumi ülesanded on sobiva raskusastmega. *

| | 1 - ei nõustu | 2 | 3 | 4 | 5 - nõustun | Ei lahendanud seda ülesannet |
|---|---|---|---|---|---|---|
| 1.1 Temperatuuri teisendamine | ○ | ○ | ○ | ○ | ○ | ○ |
| 1.2 Raha kogumine | ○ | ○ | ○ | ○ | ○ | ○ |
| 1.3 Elekter | ○ | ○ | ○ | ○ | ○ | ○ |
| 1.4 Elektriseadmed | ○ | ○ | ○ | ○ | ○ | ○ |
| 1.5 Paarisarvude filtreerimine | ○ | ○ | ○ | ○ | ○ | ○ |

Palun hinnake alljärgneva skaala alusel, kas 1. nädala ülesannete murelahendajad * arvestavad võimalike raskuskohtadega, mis tekivad ülesannete lahendamisel.

| | 1 - ei nõustu | 2 | 3 | 4 | 5 - nõustun | Ei lahendanud seda ülesannet | Lahendasin ülesande, aga ei kasutanud murelahendajat |
|---|---|---|---|---|---|---|---|
| 1.1 Temperatuuri teisendamine | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| 1.2 Raha kogumine | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| 1.3 Elekter | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| 1.4 Elektriseadmed | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| 1.5 Paarisarvude filtreerimine | ○ | ○ | ○ | ○ | ○ | ○ | ○ |

Palun hinnake alljärgneva skaala alusel, kas 1. nädala ülesannete murelahendajad aitavad ülesande lahendamisel tekkinud probleeme lahendada. *

| | 1 - ei nõustu | 2 | 3 | 4 | 5 - nõustun | Ei lahendanud seda ülesannet | Lahendasin ülesande, aga ei kasutanud murelahendajat |
|---|---|---|---|---|---|---|---|
| 1.1 Temperatuuri teisendamine | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| 1.2 Raha kogumine | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| 1.3 Elekter | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| 1.4 Elektriseadmed | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| 1.5 Paarisarvude filtreerimine | ○ | ○ | ○ | ○ | ○ | ○ | ○ |

Palun hinnake alljärgneva skaala alusel, kas 1. nädala ülesannete murelahendajad * on sellised, et ei ütle lahendust ise ette.

| | 1 - ei nõustu | 2 | 3 | 4 | 5 - nõustun | Ei lahendanud seda ülesannet | Lahendasin ülesande, aga ei kasutanud murelahendajat |
|---|---|---|---|---|---|---|---|
| 1.1 Temperatuuri teisendamine | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| 1.2 Raha kogumine | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| 1.3 Elekter | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| 1.4 Elektriseadmed | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| 1.5 Paarisarvude filtreerimine | ○ | ○ | ○ | ○ | ○ | ○ | ○ |

Kommentaarid esimese nädala ülesannete ja murelahendajate kohta.

Teie vastus

Tagasi Järgmine 2. leht 3-st Tühjenda vorm

## Nädal 2

Järgnevad küsimused puudutavad teise nädala koduülesandeid ülesandeid koos murelahendajatega.

Ülesanded:
2.1 Kursused
2.2 Identsete elementide arv
2.3 Värviliste pallide otsimise võistlus

Palun hinnake alljärgneva skaala alusel, kas 2. nädala koduülesanded keskenduvad põhiteemadele. *

| | 1 - ei nõustu | 2 | 3 | 4 | 5 - nõustun | Ei lahendanud seda ülesannet |
|---|---|---|---|---|---|---|
| 2.1 Kursused | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ |
| 2.2 Identsete elementide arv | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ |
| 2.3 Värviliste pallide otsimise võistlus | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ |

Palun hinnake alljärgneva skaala alusel, kas 2. nädala koduülesanded on selgelt sõnastatud. *

| | 1 - ei nõustu | 2 | 3 | 4 | 5 - nõustun | Ei lahendanud seda ülesannet |
|---|---|---|---|---|---|---|
| 2.1 Kursused | ○ | ○ | ○ | ○ | ○ | ○ |
| 2.2 Identsete elementide arv | ○ | ○ | ○ | ○ | ○ | ○ |
| 2.3 Värviliste pallide otsimise võistlus | ○ | ○ | ○ | ○ | ○ | ○ |

Palun hinnake alljärgneva skaala alusel, kas 2. nädala loodud koduülesanded on sobiva raskusastmega. *

| | 1 - ei nõustu | 2 | 3 | 4 | 5 - nõustun | Ei lahendanud seda ülesannet |
|---|---|---|---|---|---|---|
| 2.1 Kursused | ○ | ○ | ○ | ○ | ○ | ○ |
| 2.2 Identsete elementide arv | ○ | ○ | ○ | ○ | ○ | ○ |
| 2.3 Värviliste pallide otsimise võistlus | ○ | ○ | ○ | ○ | ○ | ○ |

Palun hinnake alljärgneva skaala alusel, kas 2. nädala koduülesannete murelahendajad arvestavad võimalike raskuskohtadega, mis tekivad ülesannete lahendamisel. *

| | 1 - ei nõustu | 2 | 3 | 4 | 5 - nõustun | Ei lahendanud seda ülesannet | Lahendasin ülesande, aga ei kasutanud murelahendajat |
|---|---|---|---|---|---|---|---|
| 2.1 Kursused | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| 2.2 Identsete elementide arv | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| 2.3 Värviliste pallide otsimise võistlus | ○ | ○ | ○ | ○ | ○ | ○ | ○ |

Palun hinnake alljärgneva skaala alusel, kas 2. nädala koduülesannete murelahendajad aitavad ülesande lahendamisel tekkinud probleeme lahendada. *

| | 1 - ei nõustu | 2 | 3 | 4 | 5 - nõustun | Ei lahendanud seda ülesannet | Lahendasin ülesande, aga ei kasutanud murelahendajat |
|---|---|---|---|---|---|---|---|
| 2.1 Kursused | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| 2.2 Identsete elementide arv | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| 2.3 Värviliste pallide otsimise võistlus | ○ | ○ | ○ | ○ | ○ | ○ | ○ |

Palun hinnake alljärgneva skaala alusel, kas 2. nädala koduülesannete murelahendajad on sellised, et ei ütle lahendust ise ette. *

| | 1 - ei nõustu | 2 | 3 | 4 | 5 - nõustun | Ei lahendanud seda ülesannet | Lahendasin ülesande, aga ei kasutanud murelahendajat |
|---|---|---|---|---|---|---|---|
| 2.1 Kursused | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| 2.2 Identsete elementide arv | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| 2.3 Värviliste pallide otsimise võistlus | ○ | ○ | ○ | ○ | ○ | ○ | ○ |

Kommentaarid teise nädala ülesannete ja murelahendajate kohta.

Teie vastus

## IV Litsents

### Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Aleksander Ontin,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose Tartu Ülikooli kursuse “Programmeerimise alused II” ülesannete ja murelahendajate loomine, mille juhendaja on Heidi Meier, reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

*Aleksander Ontin*

***15.05.2024***