TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Henri Laats

Uue ülesandekogu ja kodutöödele vastavate murelahendajate koostamine kursusele „Introduction to programming II”

Bakalaureusetöö (9 EAP)

Juhendaja: Reelika Suviste, PhD

Tartu 2023

Uue ülesandekogu ja kodutöödele vastavate murelahendajate koostamine kursusele „Introduction to programming II”

Lühikokkuvõte:

Tartu Ülikooli kursus ,,Introduction to programming II’’ on programmeerimise algkursuse teine osa. Kursus on ingliskeelne ning sellel osalevad üliõpilased on erineva tausta ja oskustasemega. Käesoleva bakalaureusetöö raames uuendati kursuse ülesandekogu ning koostati kodutöödele vastavad murelahendajad. Uus ülesandekogu aitab vähendada lahenduste levikut üliõpilaste seas ning soodustab seeläbi iseseisvat õppimist. Murelahendaja on küsimustest ja vihjetest koosnev abimaterjal, mille eesmärk on üliõpilasi toetada koduse ülesande lahendamisel, andes neile probleemide lahendamiseks suuniseid. Lisaks aitavad murelahendajad leevendada õppejõudude koormust, kuna üliõpilased ei pea abi saamiseks iga kord praktikumi juhendaja poole pöörduma. Lõputöö raames uuendati kokku 29 ülesannet ning koostati 9 murelahendajat. Loodud õppematerjalidele küsiti tagasisidet kursuse praktikumi juhendajatelt. Saadud tagasiside analüüsiti ja selle põhjal täiendati õppematerjale vastavalt soovitustele.

Võtmesõnad:

murelahendaja, programmeerimise õpetamine, ümberpööratud õpe, programmeerimise ülesanded, ADDIE mudel

CERCS: P175 Informaatika, süsteemiteooria, S270 Pedagoogika ja didaktika

Creation of a New Collection of Exercises and Troubleshooters for Homework for the Course "Introduction to programming II"

Abstract:

The University of Tartu course "Introduction to programming II" is the second part of the introductory programming course. The course is taught in English and the participating students come from various backgrounds and have different skill levels. The purpose of this bachelor's thesis was to update the existing collection of exercises and create troubleshooters for homework exercises. The new collection of exercises helps to reduce the leaking of solutions among students and therefore promotes independent learning. The troubleshooter is a help material consisting of questions and hints, which aims to support students in solving homework by giving them recommendations for solving problems. In addition, troubleshooters help to ease the workload of teachers because students do not have to ask help from the teacher every time. As part of the thesis, a total of 29 exercises were renewed

and 9 troubleshooters were made. The course practicum supervisors were asked for feedback on the created learning materials. The received feedback was analyzed and based on this, the educational materials were improved according to the recommendations.
Keywords:
troubleshooter, teaching programming, flipped classroom, programming tasks, ADDIE model
CERCS: P175 Informatics, systems theory, S270 Pedagogy and didactics

Sisukord

Sissejuhatus

Programmeerimine kogub üha enam populaarsust nii Eestis kui mujal maailmas. Sissejuhatavad programmeerimiskursused on mõeldud igale huvilisele olenemata tema varasemast taustast ja erialaga seotud kogemusest. Sellest tulenevalt on üliõpilaste oskustase nendel kursustel varieeruv. Tartu Ülikoolis on sügissemestril võimalik osaleda ingliskeelsel kursusel „Introduction to programming II", ainekoodiga MTAT.03.236, mis on jätk programmeerimise algkursuse esimesele osale. Kursus koosneb videoloengutest, praktikumidest, kodutöödest, projekti ülesandest ja eksamist. [1]. Osalemine on vabatahtlik kõigile huvilistele ning seda läbib igal aastal kuskil 50-100 üliõpilast [2].

Kursuse ülesandekogu moodustavad 20 praktikumiülesannet ja 9 koduülesannet. Õppetöö kestab viis nädalat ja igal nädalal käsitletakse praktikumides ja kodutöödes vastava nädala teemat. Käsitletavateks teemadeks on järjend, ennik, hulk, sõnastik, graafika, regulaaravaldised ja rekursioon. Ülesandekogu uuendamine aitab vähendada koduste ülesannete lahenduste levikut üliõpilaste seas ja soodustab seeläbi ülesannete iseseisvat lahendamist. Samuti on oluline, et ülesannete tekstid oleksid asjakohased ja kõnetaksid üliõpilasi, seeläbi suurendades üliõpilaste motiveeritust ülesandeid lahendama.

Tulenevalt kursusel osalevate üliõpilaste varieeruvast oskustasemest, võivad mõned osalejad vajada abi koduste ülesannete lahendamisel. Selleks on Tartu Ülikool loonud digitaalse tööriista, murelahendaja, mida on varasemalt rakendatud ka mitmetel teistel Tartu Ülikooli programmeerimise kursustel [3]. Murelahendaja on abimaterjal, mis sisaldab küsimusi ja vihjeid, et aidata ülesande lahendajal ületada keerulisi kohti, võimaldades neil lahendada ülesandeid enda valitud tempos, ilma, et nad peaksid ootama õppejõu vastust tekkinud küsimustele [3]. Selle tulemusena väheneb ka õppejõudude koormus, sest nad ei pea kulutama nii palju aega ja energiat küsimustele vastamiseks ja selgituste jagamiseks.

Käesoleva bakalaureusetöö eesmärk oli luua uus ülesandekogu ning igale kodutööle vastav murelahendaja Tartu Ülikooli kursusele „Introduction to programming II". Loodud õppematerjalid aitavad leevendada õppejõudude koormust, toetavad üliõpilasi ülesannete lahendamisel ning võivad suurendada üliõpilaste huvi õppeaine vastu, soodustades sealhulgas iseseisvat ülesannete lahendamist.

Lõputöö on jaotatud kuueks peatükiks. Esimene peatükk tutvustab kursust „Introduction to programming II", kursusel kasutusel olevat õppemetoodikat, kirjeldab kursusel olevate ülesannete tüüpe ja põhjendab uue ülesandekogu vajadust. Teises peatükis kirjeldatakse murelahendajaid, keskkonda kuhu neid loodi ning nende varasemat efektiivsust õppetegevuses. Kolmandas peatükis antakse ülevaade õppematerjalide loomise mudelist ADDIE ja selle kasutamise tõhususest õppematerjalide loomisel. Neljandas peatükis on lahti seletatud töö praktiline osa, sealhulgas kirjeldatud ADDIE mudeli rakendamist käesoleva lõputöö ülesandekogu loomisel, tagasisidestamisel ning täiendamisel. Samuti kirjeldatakse neljandas peatükis murelahendajate loomise protsessi ja analüüsitakse praktikumi juhendajatelt saadud tagasisidet. Viiendas peatükis analüüsitakse loodud õppematerjale ja võrreldakse teiste programmeerimise algkursuste õppematerjalidega. Lõputöö viimane sisupeatükk on kokkuvõte, millele järgneb lisade peatükk, mis sisaldab tehisintellekti kasutamise näidet (vt lisa 1) ja loodud ülesandekogu (vt lisa 2), murelahendajaid (vt lisa 3) ning tagasisideküsimustikke (vt lisa 4 ja lisa 5).

Käesolevas bakalaureusetöö on kasutatud rakendust Grammarly[1] ja tehisintellekti ChatGPT [2]lõputöö kirjutamisel, ülesandekogu loomisel ja murelahendajate koostamisel selleks, et eemaldada tekstidest õigekirja- ja grammatikavead ning muuta lausete sõnastust paremaks. Näiteks kasutati tehisintellekti abi, et kontrollida eelmises lõigus olnud lauset „Lõputöö viimane sisupeatükk on kokkuvõte, millele järgneb lisade peatükk (vt Lisa), mis sisaldab loodud ülesandekogu, murelahendajaid ning tagasisideküsimustikke" (vt lisa 1). Tehisintellekti rakendamise eesmärk oli lausest eemaldada õigekirjavead ja grammatikavead. Tehisintellekti vastusesse (vt lisa 1) suhtuti kriitiliselt, kuna iga soovitus ei olnud asjakohane. Näiteks oli soovitatud muuta sõnastus „lisade peatükk" kujule „lisapeatükk", mis polnud selles kontekstis sobilik. Seevastu õigekirjavea parandus sõnas „tagasisideküsimustikke" oli asjakohane.

[1] Lõputöö autor on kasutanud rakenduse Grammarly abi, et parandada ülesandekogu ülesannete tekstide loetavust. Grammarly on grammatika kontrollimise ja keeletoimetamise tööriist. Lisateavet Grammarly kohta: https://www.grammarly.com.

[2] Lõputöö autor on õigekirja ja grammatika kontrollimiseks ning teksti loetavuse parandamiseks rakendanud lõputöös, loodud ülesandekogus ja murelahendajates ChatGPT abi (2023), s.o kasutades keelemudelit, mille väljaõpe põhineb suurel hulgal erinevatel tekstiallikatel. ChatGPT on välja töötatud OpenAI poolt. Lisateavet ChatGPT ja OpenAI kohta: https://openai.com. Mudelile esitatud sisend: "Kontrolli õigekirja ja grammatikat: *kontrollitav tekst*"

1. Kursus „Introduction to programming II”

Järgnevas peatükis antakse ülevaade kursusest ,,Introduction to programming II”, selle sisust, struktuurist ja hindamissüsteemist. Lisaks kirjeldatakse kursusel kasutusel olevat ümberpööratud klassiruumi õppemeetodikat, kursusel kasutusel olevaid ülesannete tüüpe ning põhjendatakse uue ülesandekogu loomise vajalikkust.

1.1 Kursuse ülevaade ja korraldus

Tartu Ülikooli kursus ,,Introduction to programming II” on jätk kursuse esimesele osale, mille eesmärgiks on õpetada algteadmised programmeerimisest [1]. Kursusel on kasutusel programmeerimiskeel Python, mis on TIOBE indeksi järgi kõige populaarsem programmeerimiskeel maailmas [4]. Ühtlasi on Python suurepärane keel algajatele programmeerimise õpetamiseks oma lihtsasti arusaadava süntaksi ja loogilise struktuuriga, mis teevad koodi lugemise ja kirjutamise kergeks [5]. Erinevalt kursuse esimesest osast on jätkuosa kõigile valikuline ning sellest tulenevalt ka väiksema osalejate arvuga. Varasematel aastatel on kursusel osalenute arv jäänud vahemikku 50 kuni 100 üliõpilast [2]. Õppetegevus kursusel on inglise keeles, olles suunatud välisüliõpilastele või eesti keelt mitte kõnelevatele inimestele. Eesti keelt kõnelevatele üliõpilastele on Tartu Ülikoolis olemas analoogne kursus nimega ,,Programmeerimine”. Erinevus eestikeelse kursusega on vaid selles, et kursusel ,,Programmeerimine” on esimene ja teine osa kokku pandud [6].

Õppetöö kestus kursusel ,,Introduction to programming II” on 5 nädalat, pärast mida tuleb üliõpilastel sooritada eksam. Nende nädalatega üritatakse üliõpilastele edasi anda teadmised peamistest andmetüüpidest ja andmestruktuuridest ning oskuse kasutada põhilisi programmeerimise konstruktsioone ja standardoperatsioone. Samuti on oluline, et kursuse lõpuks suudab üliõpilane enda programmi analüüsida, töökäiku selgitada ning oma koodilõiku testida ja täiendada [1].

Kursuse õppetöö koosneb videoloengutest, koduülesannetest, praktikumi ülesannetest, projektiülesandest ja eksamist. Iga tegevus on hinnatud, millest peamise osa lõpphindest moodustab eksam 56 punktiga võimalikust 101-st punktist. Projekti loomise eest on võimalik saada 20 punkti. Ülejäänud punktisumma kogub üliõpilane kursuse vältel lahendades loengute- ja tekstiliste materjalide kohta käivaid teste, lahendades koduülesandeid ja osaledes praktikumides. Kursuse lõpphinne on eristav, kasutatades Tartu Ülikoolis kasutusel olevat hindamissüsteemi A, B, C, D, E, F või mi [1]. Vaatamata sellele, et punktide hulk pole iga

ülesande juures suur, on iga tegevus oluline uue informatsiooni omandamisel ja kinnistamisel.

1.2 Ümberpööratud klassiruum

Ümberpööratud klassiruum (ingl *flipped classroom*) on õppemeetod, mille eripäraks on iseseisev töö, mida üliõpilased peavad tegema enne õppejõuga praktikumi. Üliõpilastel tuleb endale selgeks teha vastava nädala teemaga seotud õppematerjalid ning praktikumis lahendatakse teemakohaseid ülesandeid [7]. Selline õppemeetod on laialt levinud Tartu Ülikoolis, olles kasutusel nii kursusel „Introduction to programming II" kui ka mitmetel teistel programmeerimise kursustel [2]. Ümberpööratud klassiruumi õppemeetodil on mitmeid positiivseid omadusi. Esiteks on üliõpilasel võimalik õppida omas tempos, vabalt valitud ajal ja keskenduda arendamist vajavatele aspektidele (teadmised ja oskused). Lisaks aitab see tõsta üliõpilaste kaasatust tundides, kuna üliõpilased on teemaga juba eelnevalt kursis. Ühtlasi saavad nad ülesannete lahendamisel õppejõult koheselt abi küsida, sest nad ilmselt juba teavad, mida küsida või vähemalt tunnevad teemakäsitluses olevaid mõisteid. Lisaks väheneb õppejõudude koormus, kuna füüsilistele tundidele kulub vähem aega [7-8]. Peamine oht ümberpööratud õppemeetodi puhul esineb olukorras, kus üliõpilased ei ole teinud iseseisvat tööd ja sellest tulenevalt ei pruugi nad praktikumis lahendatud ülesannetest aru saada ning ei omanda vastava nädala teemades taotletud oskusi [8-9].

Tavaline nädal kursusel „Introduction to programming II" osaleva üliõpilase jaoks algab iseseisvate õppematerjali läbitöötamisega, mis koosneb veebikursuse videoloengutest ja tekstilistest materjalidest. Veebikursus, mille videoloenguid üliõpilased vaatavad enne tundi on ingliskeelne programmeerimise algkursuse Pythoni programmeerimiskeeles, milles käsitletakse samu teemasid, mida õpetatakse "Introduction to programming II" kursusel [1, 10]. Videoloengud ja tekstiline materjal on õpetamise juures olulised, et üliõpilaste jaoks lahti seletada nädala põhiteema, tuues seejuures asjakohaseid näiteid [7].

Pärast iseseisvat materjali läbitöötamist on üliõpilasel vaja lahendada teemakohane kodune ülesanne [1]. See on programmeerimise õppimiseks üks efektiivsemaid meetodeid, sest õppija peab reaalselt arusaama kontseptsioonidest ja olema võimeline iseseisvalt probleemidele lahendust leidma [11]. Pärast kodutööd toimub õppejõuga praktikum, mille peamine eesmärk on kinnistada nädala teemat, lahendades koos õppejõuga keerukamaid teemakohaseid ülesandeid. Samuti saavad üliõpilased praktikumis küsida nõu ja täiendavaid

selgitusi [7]. Nii kodutööde kui ka praktikumi ülesannete arv on nädalati varieeruv, kuid keerukuselt on kodused ülesanded oma teemakäsitluses ning struktuurilt lihtsamad ja jõukohasemad kui mõned praktikumi ülesanded [1].

Ümberpööratud klassiruumi õppe efektiivsust toetavad mitmed uuringud. Üheks näiteks on Nwosisi, Ferreira, Rosenbergi ja Walshi [12] läbiviidud uuring, milles jõuti järeldusele, et ümberpööratud klassiruumi õppemeetod aitab õpilastel saavutada paremaid tulemusi, tuues põhjenduseks varasemast kõrgema keskmise punktisumma. Uuringus osalesid kaks kursust, mis kasutasid ümberpööratud klassiruumi õppemetoodikat. Õpilaste tulemusi võrreldi eelneva aasta kursustega. Mõlemal kursusel paranes keskmine punktisumma õpilaste seas üle 2,6% ning samuti vähenes kursuse mittelõpetamise osakaal. Sarnase uuringu viisid läbi Amresh, Carberry ja Femiani [13], kes hindasid ümberpööratud klassiruumi tõhusust arvutiteaduse tudengite õpetamisel. Selles uuringus jõuti samuti järeldusele, et ümberpööratud õppemeetodi rakendamine aitab tõsta õpilaste tulemusi. Samas selgus ka, et osade õpilaste jaoks käib selline lähenemine üle jõu suure iseseisva töö tõttu. Taşpolat, Özdamli ja Soykan [8] on oma uuringus toonud nõrkuseks veel olukorrad, kus üliõpilane ei ole teinud iseseisvat tööd või iseseisva õppe tõttu puudub tal võimalus koheselt õppejõult abi küsida ning seetõttu jääb tal informatsioon omandamata. Positiivsete omadustena on Taşpolat, Özdamli ja Soykan oma uuringus välja toonud veel eelnevale lisaks, et õpilased, kes õppisid ümberpööratud klassiruumi järgi olid positiivsemalt meelestatud programmeerimisse, iseseisvamad ning oskasid enda aega paremini planeerida [8]. Kokkuvõtlikult on võimalik väita, et hästi korraldatud ümberpööratud klassiruumi õppemeetod aitab suurendada lõpetajate osakaalu, motiveeritust ja parandada õpitulemusi.

1.3 Kursuse „Introduction to programming II” ülesannete tüübid

Kursuse „Introduction to programming II’’ läbimiseks tuleb üliõpilasel lahendada iganädalased testid ja ülesanded, esitada iseseisvalt tehtud projektitöö ja sooritada eksam [1]. Järgnevalt antakse ülevaade kursuse varasemas ülesandekogus esinenud ülesannete tüüpidest.

Ülesannete tüüpide määramisel on aluseks võetud 2021. aastal Ruth Schihalejevi magistritöö raames loodud veebileht [14-15]. Ruth Schihalejevi on jaganud programmerimisülesanded kolme peamisesse gruppi, mis koosnevad 11 erinevast alamkategooriast. Ülesannete grupeerimine ja kategoriseerimine on järgnev [14]:

- Tea ja mõista - Baasteadmiste tundmine ning nende põhjal koodist arusaamine.
    - Tuvasta
    - Kontrolli
    - Selgita
- Rakenda ja anlüüsi - Olemaseolevate oskuste rakendamine keerulisemates ülesannetes
    - Seosta
    - Teosta
    - Kohanda
    - Transleeri
- Sünteesi ja hinda - Lahendajalt oodatakse sügavamat ettemõtlemist ja loomingulisust
    - Silu
    - Rakenda
    - Modelleeri ja projekteeri
    - Uuenda

Kursuse „Introduction to programming II" videoloengutes ja lugemismaterjalides õpetatakse üliõpilastele käsitletavate teemadega seotud teoreetilisi teadmisi ja tuuakse illustreerimiseks mõned lihtsamad näited, kattes seeläbi teemakohaste esimeste tasemete ülesannete tutvustamise. Praktikumi ülesannetes ja kodustes ülesannetes lahendatakse sellest natukene keerulisemaid ülesandeid, kus üliõpilane peab kirjutama etteantud juhiste järgi ülesande probleemi lahendava koodilõigu. Sedasorti ülesanded liigituvad Schihalejevi veebilehe alusel „Rakenda ja analüüsi'' gruppi, mida on kirjeldatud kui grupina, kuhu liigituvad ülesanded, kus rakendatakse olemasolevaid teadmisi, et keerulisemates ülesannetes koodist aru saada, seda ise kirjutada või modifitseerida [14]. Sealjuures on nende ülesannete puhul ette antud selged juhised. Enamjaolt kuuluvad kursusel olevad ülesanded „Teosta'' kategooriasse, mida on kirjeldatud järgnevalt: „kirjuta etteantud juhiste järgi programmi kood''. Samuti leidub ülesandeid, mida on sobilik liigitada „Kohanda'' kategooriasse [14]. Sellisteks on ülesanded, milles tuleb varasemalt kirjutatud koodi modifitseerida, et lahendus vastaks uue ülesande kirjeldusele, muutes sealjuures koodi konstruktsiooni ja andmestruktuure.

Kursuse iganädalaste testide, näiteülesannete, kursuse lõpus toimuva eksami ja projektitöö hulgas on ülesandeid, mis kuuluvad oma ülesande tüübi poolest grupi „Tea ja mõista" alamkategooriatesse, kuid käesoleva lõputöö raames neid ei käsitleta.

1.4 Uue ülesandekogu vajadus

Selleks, et üliõpilased saaksid kursusest maksimaalselt kasu, on oluline, et nad panustaksid enda aega teemadesse süvenemiseks ja lahendaksid iseseisvalt ülesandeid. Sellest tuleneb vajadus luua uusi õppematerjale, kuna materjalide aktuaalsus ja seostatavus üliõpilaste huvidega, hobidega ja teiste meelepäraste teemadega motiveerib neid paremini õppima [16]. Samuti on õppematerjalide uuendamine vajalik, et vähendada ülesannete lahenduste levimist üliõpilaste seas ning seeläbi soodustada iseseisvat õppimist. Iseseisvalt ülesandeid lahendades peab üliõpilane lahendama mitmesuguseid probleeme, mis on ühtlasi efektiivne viis programmeerijana arenemiseks, kus suur osa tööst on samuti probleemide lahendamine [11].
Kursuse esimese osa „Introduction to programming" (MTAT.03.236) õppematerjale uuendati eelmisel aastal (2022), kui Taniel Saareveti bakalaureusetöö raames loodi kursusele uued koduülesanded ja nendele vastavad murelahendajad [17]. Käesoleva lõputöö raames uuendati kursuse teise osa ülesandekogu (nii kodu- kui ka praktikumiülesanded) ja koostati kodustele ülesannetele vastavad murelahendajad. Tulenevalt sellest, et praktikumide ülesandeid lahendatakse praktikumides koos praktikumi juhendajaga, puudus vajadus luua nendele ülesannetele vastavad murelahendajad, sest üliõpilased saavad abi küsida õppejõult. Kokku uuendati 20 praktikumi ülesannet, 9 koduülesannet ning koostati 9 murelahendajat.

2. Murelahendajad õppetöös

Selles peatükis kirjeldatakse murelahendajate olemust ja nende kasutamist programmeerimise õpetamiseks. Samuti antakse ülevaade murelahendajate rakendamise tõhususest teistel programmeerimise kursustel ja tuuakse välja võimalikud kasutegureid tehtud uuringute põhjal.

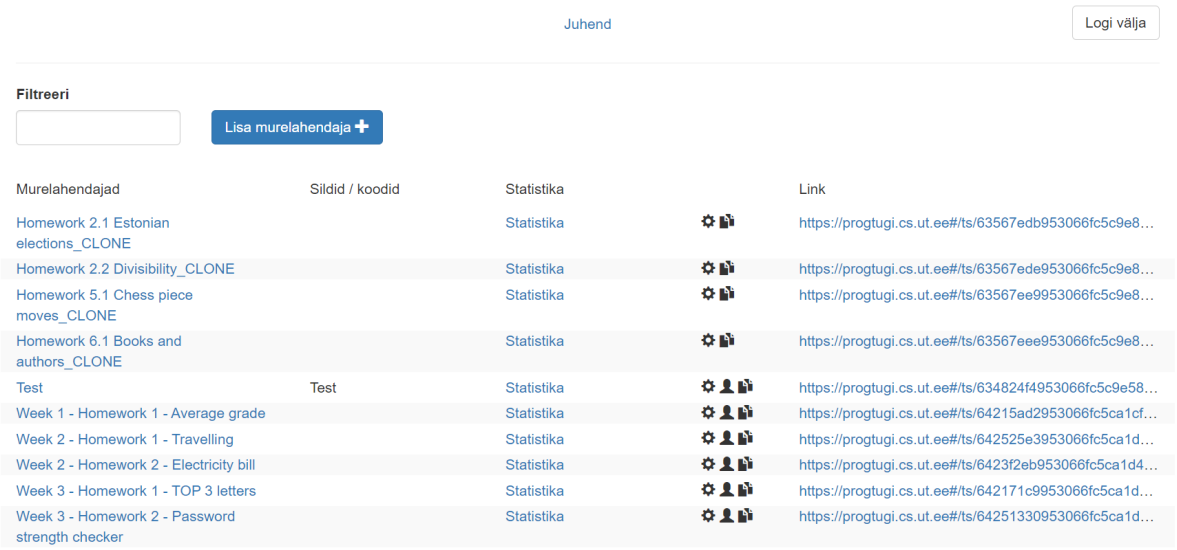
2.1 Murelahendaja olemus

Vigade leidmine (ingl *troubleshooting*) on levinud probleemide lahendamise viis, milles kõigepealt tuvastatakse vea põhjus, mis seejärel lahendatakse. Tavaliselt kasutatakse vigade leidmise meetodit elektroonikas, tarkvaras ja mehaanikas, kuid seda võib rakendada ka peaaegu igas teises valdkondades [18]. Vigade leidmine tarkvaras võib osutuda eriti keeruliseks, sest probleemid võivad olla põhjustatud mitmest erinevatest veast.

Üks võimalik vigade leidmise viis on süsteemne lähenemine, kus vaadatakse võimalikud probleemi põhjused ükshaaval läbi ning seejärel elimineeritakse need, kuni probleem on lahendatud. Sellisele süsteemile tugineb Tartu Ülikooli abisüsteem Murelahendaja (ingl *troubleshooter*), mille eesmärk on toetada üliõpilast ülesande lahendamisel [3]. Käesolevas lõputöös väljatöötatud murelahendajad aitavad kasutajal üle vaadata kõikvõimalikud probleemsed kohad ning annavad talle vajadusel vihjeid, kuidas probleemi lahendada. Abisüsteem on loodud küsimuste ja vihjete vormis, milles uuritakse, kas kasutaja on erinevate etappide loomisega hakkama saanud. Kui kasutaja pole mõne etapi loomisega hakkama saanud, siis talle antakse selle etapi lahendamiseks vastavad suunised.

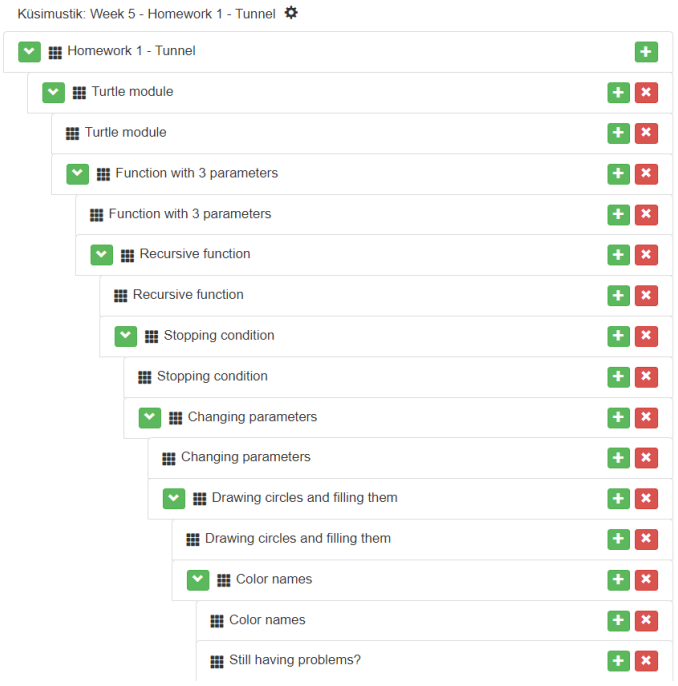
2.2 Murelahendajate keskkond

Käesolevas lõputöös valiti murelahendajate loomise keskkonnaks Tartu Ülikooli keskkond Progtugi [19]. Tartu Ülikooli murelahendajate keskkond loodi 2016. aastal Vello Vaherpuu bakalaureusetöö raames [20]. Keskkond on eestikeelse kasutajaliidesega ning sisaldab kahte vaadet: kasutajavaadet (joonised 1, 2 ja 3) ning külalisvaadet (joonised 4, 5 ja 6). Kasutajavaates (joonisel 1) saab koostada murelahendajaid, jagada neid teiste kasutajatega ning jälgida murelahendajate kasutamise statistikat.



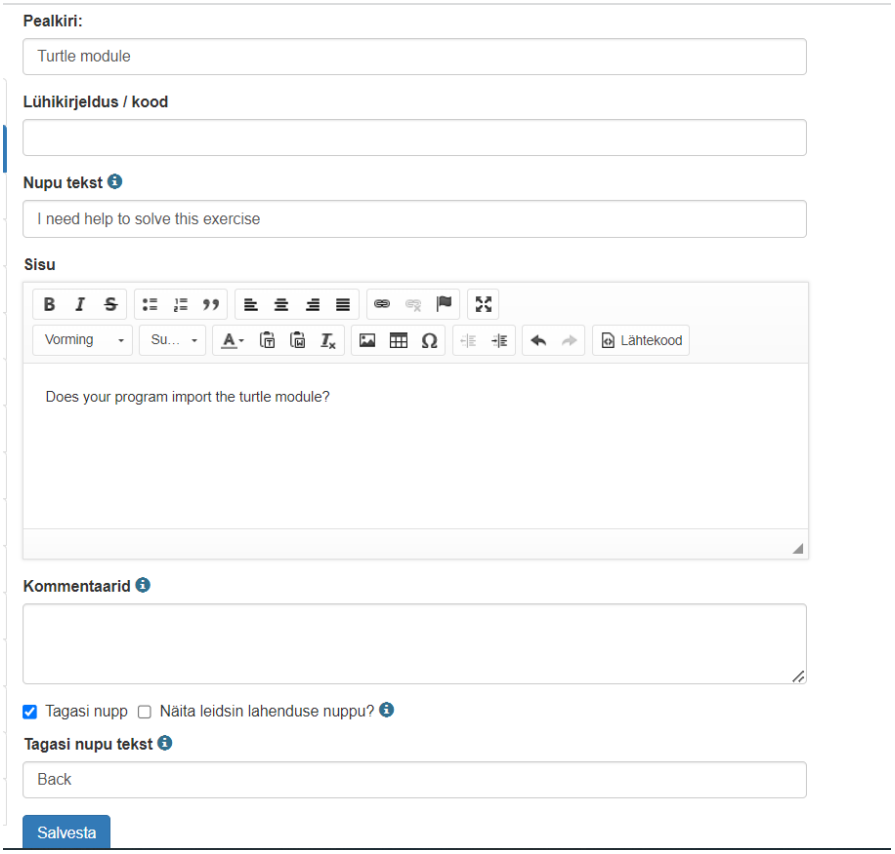
Joonis 1. Kasutajavaate avamenüü

Alloleval joonisel (Joonisel 2) on kujutatud murelahendajat kasutajavaates, mis ilmneb murelahendaja loomisel või redigeerimisel. Murelahendaja struktuur sarnaneb puuga, kus iga küsimus võib hargneda mitmes suunas. Igas murelahendajas on esmalt esitatud ülesande tekst, millele järgneb esimene küsimus. Iga küsimuse all on kaks valikut: üks viib probleemi lahendamise vihjeni ja teine järgmise küsimuse juurde. Murelahendaja lõppu jõudes soovitatakse üliõpilastel pöörduda praktikumi juhendaja poole, kui probleemidele lahendust ei leitud. Murelahendaja muutmiseks ja täiendamiseks saab kasutada pluss-nuppu, et lisada uusi harusid ehk küsimusi või risti-nuppu, et eemaldada haru ehk küsimust.



Joonis 2. Murelahendaja struktuur

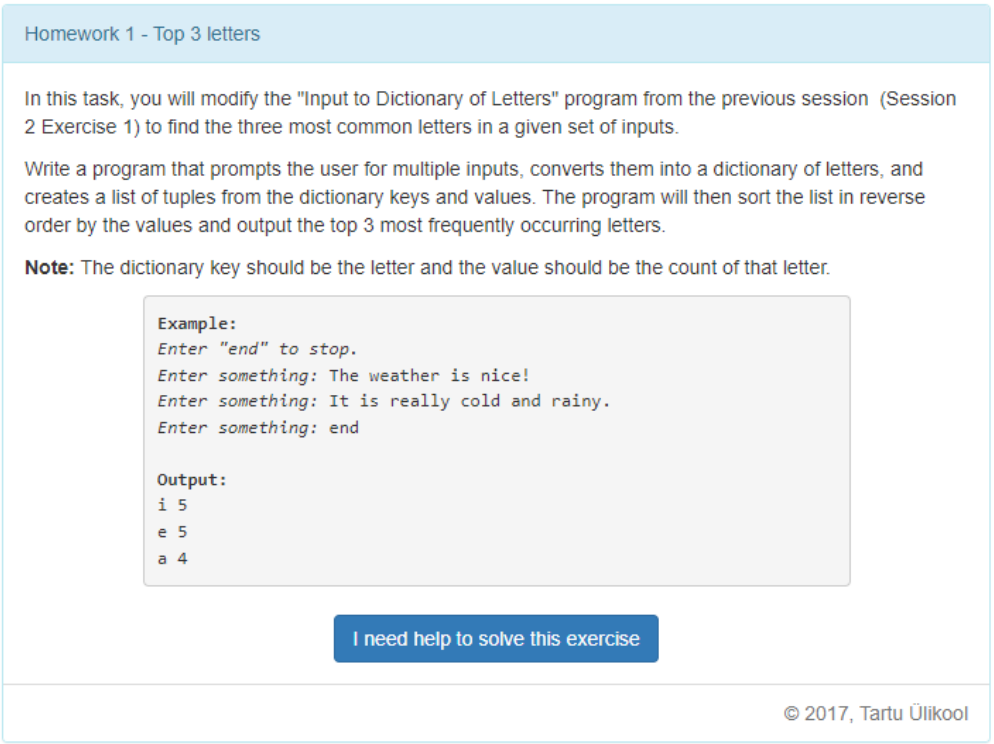
Joonisel 3 on kujutatud murelahendaja küsimuse vormi, mille abil luuakse või muudetakse konkreetset küsimust.



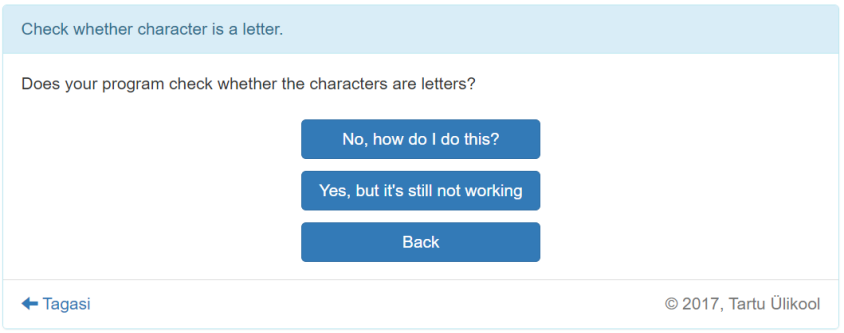
Joonis 3. Murelahendaja küsimuse vorm

Külalisvaates on murelahendajaid võimalik kasutada vaid vihjete saamiseks ning muude tegevuste (koostamine, statistika) jaoks võimalused puuduvad.

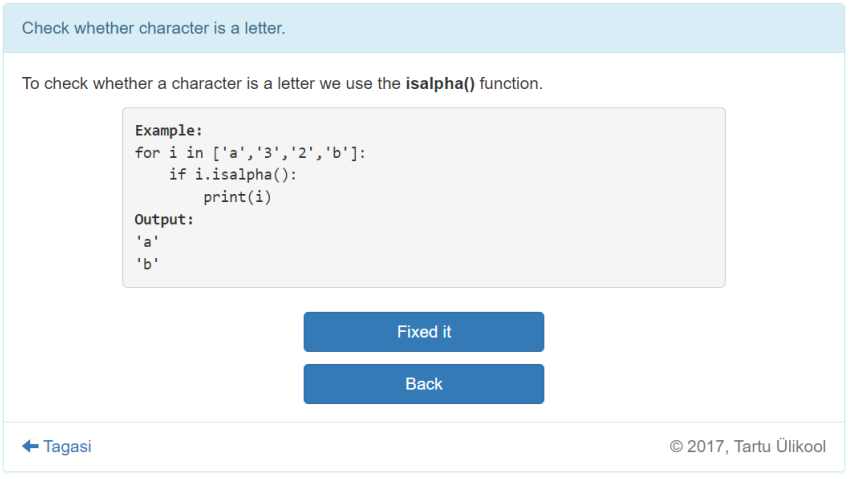
Joonistel 4 kuni 6 on kujutatud külalisvaadet erinevates etappides. Esmalt avaneb kasutajale ülesande kirjeldus, millele järgnevad ülesandega seotud küsimused ning lahendustele suunavad vihjed.



Joonis 4. Ülesande kirjeldus



Joonis 5. Murelahendaja küsimus



Joonis 6. Probleemi lahendamise suunis

Käesoleva lõputöö loomise hetkeks (kevad 2023) on kuus Tartu Ülikool arvutiteaduse instituudi üliõpilast oma lõputöö raames loonud Progtugi keskkonda murelahendajaid [21]. Üks nendest üliõpilastest on Taniel Saarevet, kelle bakalaureusetöös uuendati koduseid ülesandeid ja loodi murelahendajad kursuse „Introduction to programming" esimesele osale [17]. Kursuse jätkule lõi käesoleva lõputöö autor uue ülesandekogu ja kodutöödele vastavad murelahendajad. Murelahendajate loomise protsessi on kirjeldatud neljandas peatükis (vt ptk 4).

2.3 Murelahendaja tõhusus

Esmakordselt rakendati Tartu Ülikooli murelahendajate keskkonda 2016. aastal kursusel „Programmeerimisest maalähedaselt" [22]. Tegemist on MOOCiga (ingl *Massive Open Online Courses*) ehk suure osalejate arvuga kursusega [2-3]. Kursuse tagasisidest selgus, et „Programmeerimisest maalähedaselt'' õppejõududele esitatud küsimuste arv vähenes 29% ning 80,2% osalejatest kasutasid murelahendajaid, kellest 90% said abi [3]. Joosep Kaimre bakalaureusetöös [23] loodud murelahendajad kursusele „Programmeerimine'' osutusid samuti efektiivseks. Kaimre on tagasisidele tuginedes välja toonud, et üliõpilastest 86,4% kasutas murelahendajaid ning enam kui 90% neist sai ka abi. Positiivset tagasisidet murelahendajatele sai ka Anett Klaanberg enda 2020. aastal tehtud bakalaureusetöös, kus ta koostas murelahendajaid kursusele „Objektorienteeritud programmeerimine" [24].

Murelahendajate koostamise keerukust võib pidada abimaterjalide nõrgaks küljeks, kuna probleemsete kohtade tuvastamine, küsimuste-vihjete komplektide loomine ja Progtugi keskkonda sisestamine võib olla aeganõudev ja keeruline. Lisaks võib murelahendajate kasutamisel esineda teine negatiivne külg, kus need soodustavad üliõpilastes laiskust läbi selle, et üliõpilased suunduvad koheselt murelahendaja poole ning ei ürita ise lahenduseni jõuda. Seevastu Klaanbergi [24] uurimusele tuginedes üritavad üliõpilased esmalt ülesannet ise lahendada ning murelahendajat kasutatakse hätta sattumisel või selleks, et näha murelahendaja soovitusi teadmiste suurendamiseks. Kokkuvõtlikult võib väita, et hästi koostatud murelahendajad aitavad tõsta üliõpilaste tulemusi ja vähendavad õppejõudude koormust.

3. Õppematerjalide loomise mudel ADDIE

Õppematerjalide tõhusaks loomiseks on olemas erinevad mudelid. Levinumateks mudeliteks on [25]:

- ADDIE mudel,
- Merrilli juhendamise põhimõtted,
- Gagne'i juhiste üheksa sündmust
- Bloomi taksonoomia

Käesolevas bakalaureusetöös koostati uus ülesandekogu kursusele „Introduction to programming II" järgides ADDIE mudeli põhimõtteid. ADDIE mudel valiti seetõttu, et see pakub õppematerjalide loomiseks struktureeritud süsteemi, hõlmates kõiki olulisi faase varasemate materjalide analüüsimisest kuni loodud materjalide hindamiseni. Järgnevas peatükis kirjeldatakse ADDIE mudelit ning selle positiivseid ja negatiivseid külgi. Neljandas peatükis (vt ptk 4) kirjeldatakse ADDIE mudeli etappide rakendamist selles lõputöös.

3.1 ADDIE mudel

ADDIE mudel on üks populaarsematest õppematerjalide loomise mudelitest, olles laialt levinud nii tarkvaraarendajate kui ka õppejõudude seas [26]. ADDIE mudeli nimi on akronüüm mudeli sammudest, mis aitavad materjalide koostamist kavandada ja loodud materjale edukalt rakendada. Need viis sammu on inglise keeles järgnevad [26]:

- ***A****nalyse* - analüüsimise etapp
- ***D****esign* - kujundamise/kavandamise etapp
- ***D****evelop* - materjali loomise/arendamise etapp
- ***I****mplement* - loodud materjali rakendamise etapp
- ***E****valuate* - etapp, kus hinnatakse loodud materjale peale kasutamist

Mudeli sammud võivad näiliselt üksteisega aegajalt kattuda, kuid ometi pakuvad need erinevaid suuniseid materjali väljatöötamiseks.

Imed Bouchrika [25] on välja toonud ADDIE mudeli heaks omaduseks selle sobivuse erinevates õppeviisides nagu e-õpe, kohapealne õpe ja hübriidõpe. Lisaks soodustab ADDIE mudeli kasutamine analüüsi ning aitab kursuse loojatel hinnata õppematerjalide vajadusi ja tulemuslikkust peaaegu kõigis etappides. Mudeli lihtne struktuur muudab selle kasutamise lihtsaks ja tõhusaks. Iga tegevus on kindlalt määratletud ja sellel täitmiseks on vaja läbida kindel protsess. Samuti on Bouchrika välja toonud, et kuigi ADDIE mudeli lihtsus on selle

üks eelistest, võib see ühtlasi olla ka mudeli puuduseks, kuna mudeli iga etapi läbimine võib olla ajakulukas ja seetõttu osutuda ebaefektiivseks.

ADDIE mudel on efektiivne peamiselt suurte ja keeruliste kursuste loomisel, kuna mudel pakub struktureeritud lähenemist õppematerjalide väljatöötamisele. Samas ei pruugi ADDIE mudel olla kõige sobivam kursustel, kus õpetaja ja õpilase vaheline suhtlus on oluline [25].

3.2 ADDIE mudeli etappide kirjeldused

ADDIE mudeli esimene etapp on analüüsimine, mis on ühtlasi kõige olulisem samm kogu protsessis. Selles etapis tuleb tuvastada potentsiaalsed murekohad, määrata soovitud lõppeesmärgid ja kindlaks teha õppijate varasemad teadmised ning oskused õpetatavas valdkonnas [26]. Analüüs tagab loodavate õppematerjalide sisulise ja keerukuse vastavuse üliõpilaste oskustasemega, kes neid hiljem kasutama hakkavad. Analüüsi etapi efektiivsuse tõstmiseks on võimalik jagada see etapp alamülesanneteks, mis hõlmavad järgmist [27]:

- Loodavate õppematerjalide kasutajate tuvastamine ja nende oskuste määramine, et õppematerjalid sisaldaksid vajalikku informatsiooni. Välja jäetakse üleliigne informatsioon, mida üliõpilased peaksid juba eelnevalt teadma.
- Oskuste määramine, mida üliõpilased peaksid kursuse õppematerjalidest omandama, et lihtsustada materjalide koostamist ja tagada parem efektiivsus.
- Tegevuste määramine, et saavutada lõppeesmärgid võimalikult efektiivselt ja väikese ajakuluga.
- Tulemuste ja lävendite määramine, et hinnata materjalide efektiivsust.

Pärast analüüsi etappi järgneb kavandamine, mille eesmärk on kindlaks määrata, kuidas saavutada soovitud tulemused läbi loodavate õppematerjalide [28]. Kavandamise etapis kasutatakse eelnevas etapis kogutud andmeid, et koostada plaan õppematerjalide loomiseks, et need vastaksid õppe-eesmärkidele [26]. See etapp on oluline, kuna struktureeritud viisil materjalide koostamine tagab kiirema ja parema lõpptulemuse.

Arenduse etapp on ADDIE mudeli kolmas etapp, mille edukus sõltub eelnevate etappide kvaliteedist [28]. Kui analüüsi- ja kavandamise etapid on läbi viidud tõhusalt ning paika on pandud kindel plaan, siis peaks arendusfaas sujuma kiiresti. Vastasel juhul võib juhtuda, et arenduse etappi tuleb korduvalt läbida. Arendusetapis määratakse sobivad tehnoloogiad, mida

kasutatakse õppematerjalide loomisel ja nende rakendamisel [27]. Etapi lõpuks valmivad uued õppematerjalid koos sobivate tehnoloogiatega, mida saab kasutada vastavalt eelnevalt kindlaks määratud plaanile ja eesmärkidele.

Rakenduse etapp on ADDIE mudeli neljas etapp ja selles sammus rakendatakse loodud õppematerjale esimest korda testgrupi peal. Selles etapis on oluline, et materjalide sisu oleks esitatud arusaadavalt nii õppejõududele kui ka õpilastele ning vajalikud tehnoloogiad oleksid lihtsasti kättesaadavad [26]. Nada Aldoobie on rõhutanud, et selles etapis on peamiseks eesmärgiks tagada õppematerjalide tõhus kasutamine [27]. See hõlmab selgeid ülesandeid ja juhiseid vajalike tehnoloogiate installimiseks ja kasutamiseks. Rakenduse etapis saab esimese hinnangu materjalide kasutamise ajakulust ja võimalikest probleemidest.

ADDIE mudeli viimane etapp on hindamine, milles küsitakse tagasisidet loodud materjalide kohta ja neid vajadusel täiendada, et tagada nende efektiivsus ja kasulikkus üliõpilastele [26]. Hindamist saab teha kahe erineva tüübi järgi: pidev hindamine, mis hindab saavutatud tulemusi igas etapis ning kokkuvõtlik hindamine, mida kogutakse materjalide kasutajatelt [27]. Täpne hindamine aitab kindlaks teha, kas seatud eesmärgid on saavutatud ning kuidas tulemusi saaks tulevikus veelgi täiendada.

4. Õppematerjalide koostamise protsessi kirjeldus ja tagasiside analüüs

Allolevas peatükis kirjeldatakse ADDIE mudeli rakendamist uue ülesandekogu loomisel ning selgitatakse koduülesannetele vastavate murelahendajate koostamist. Samuti antakse ülevaade tagasiside kogumise protsessist nii ülesandekogule kui ka murelahendajatele ning kirjeldatakse saadud tagasiside analüüsimist.

4.1 Ülesandekogu loomine järgides ADDIE mudelit

Lõputöös lähtuti uue ülesandekogu loomisel ADDIE mudelist. ADDIE mudeli kirjeldusega on võimalik tutvuda käesoleva töö kolmandas peatükis (vt ptk 3). Vastavalt mudelile oli ülesandekogu koostamisel esimeseks sammuks analüüsietapi läbimine. Käesolevas lõputöös hõlmas analüüsietapp kursuse varasema õppematerjaliga tutvumist, mille käigus fikseeriti kursuse iganädalased teemad ja osaoskused. Samuti uuriti kursuse esimest osa selleks, et kaardistada programmeerimisalased teadmised, mis üliõpilastel võiksid kursuse alguses olemas olla. Kursuse „Introduction to programming'' esimesest osast peaksid üliõpilased omama teadmisi järgmistes teemades: muutujad ja avaldised, tingimuslaused, funktsioonid, iteratsioonid, sõned, failid, hulgad ja graafika [1]. See informatsioon oli oluline seetõttu, et loodud ülesanded ei sisaldaks teadmisi, mis võivad üliõpilastele osutuda liiga keeruliseks. Ühtlasi oli oluline, et ülesannete tekstides oleks uus informatsioon eraldi välja toodud, kui seda loengu materjalides pole kajastatud.

Iga uue ülesande loomisel määrati varasemalt kasutusel olnud ülesande sisu ja keerukus, et uued ülesanded ei erineks keerukuselt ega teemakäsitluse poolest. Loodud ülesannete sisud tehti võimalikult eluliseks, et need kõnetaksid üliõpilasi ning seeläbi oleksid üliõpilased motiveeritud neid lahendama [16]. Samuti on üheks analüüsietapi osaks eesmärkide seadmine. Lõputöö raames oli selleks eesmärgiks asjakohase, teemakohase ja keerukuselt vastava ülesandekogu loomine.

ADDIE mudeli teises etapis ehk kavandamise etapis koostati eelmises etapis saadud informatsiooni põhjal plaan õppematerjalide loomiseks. Ülesannete koostamiseks otsustati kasutada Google Colabi keskkonda, kuna selles on mugav kirjutada nii ülesande teksti kui ka võimaliku lahenduse koodilõiku. Pärast ülesandekogu valmimist sisestati loodud ülesanded (ülesannete tekstid) Google Docsi dokumenti, et loodud ülesandekogu saaks mugavalt praktikumi juhendajatele tagasisidestamiseks näidata. Vastavalt eesmärgile luua

ülesandekogu, mis sisaldab praktilisi ülesandeid ning võimaldab õppijatel oma oskusi rakendada, oli oluline arvestada sellega, et erinevate oskustasemetega õppijad vajavad erineva keerukusega ülesandeid. Seetõttu sisaldab uus ülesandekogu nii lihtsamaid ülesandeid, mis sobivad algajatele ja aitavad neil teemadest aru saada, kui ka keerukamaid ülesandeid, mis panevad edasijõudnute oskused proovile. Materjalide keeleliseks toimetamiseks kasutati rakendust Grammarly[3] ja tehisintellekti ChatGPT[4]. ChatGPT-d kasutati ülesandekogu loomisel selleks, et ülesande tekstidest eemaldada grammatika- ja kirjavead ning küsida tagasisidet ülesande sõnastusele, et vajaduse korral lausete struktuuri täiendada. See tegevus parandas loodud ülesandekogu kvaliteeti ning aitab ennetada võimalikke sõnastusest tulenevaid arusaamatusi tulevaste õppematerjalide kasutajate seas.

ADDIE mudeli kolmas etapp on arendusfaas. Käesolevas lõputöös loodi selles etapis ülesandekogu tekstid kursuse nädalate ehk teemade kaupa. Kõik ülesanded toimetati keeleliselt üle, pärast mida koostati tagasiside küsimustik (vt lisa 4). Pärast tagasiside saamist täiendati ülesandekogu (vt lisa 2) vastavalt soovitustele. Ülesandekogu loomine kestis 2022. aasta oktoobrist kuni 2023. aasta aprilli keskpaigani. Pärast ülesandekogu loomist järgnes murelahendajate koostamine.

Lõputöö raames ei olnud võimalik ADDIE mudeli neljandat etappi ehk materjalide rakendamist üliõpilaste peal läbi viia, kuna kursus toimub järgmine kord alles 2023. aasta sügissemestril. Seetõttu kaasati praktikumi juhendajad, kes aitasid ülesannete tagasisidestamisel, töötades läbi ülesanded ja tuues välja puudused vastates küsimustikule. Saadud tagasiside analüüsiti ning loodud ülesandekogu täiendati vastavalt soovitustele. Tagasiside põhjal viidi läbi ka materjalide hindamine ehk teostati ADDIE mudeli viimane etapp.

Materjalide hindamiseks kasutati õppematerjali kvaliteedi hindamismudelit LORI (ingl *Learning Object Review Instrument*). Mudeli järgi on võimalik hinnata õppematerjali kvaliteeti läbi ekspertide (selles töös praktikumi juhendajad) antud hinnangu. LORI mudel

[3] Lõputöö autor on kasutanud rakenduse Grammarly abi, et parandada ülesandekogu ülesannete tekstide loetavust. Grammarly on grammatikakontrollimise ja keeletoimetamise tööriist. Lisateavet Grammarly kohta: https://www.grammarly.com.

[4] Lõputöö autor on õigekirja ja grammatika kontrollimiseks ning teksti loetavuse parandamiseks rakendanud lõputöös, loodud ülesandekogus ja murelahendajates ChatGPT abi (2023), s.o kasutades keelemudelit, mille väljaõpe põhineb suurel hulgal erinevatel tekstiallikatel. ChatGPT on välja töötatud OpenAI poolt. Lisateavet ChatGPT ja OpenAI kohta: https://openai.com. Mudelile esitatud sisend: "Kontrolli õigekirja ja grammatikat: *kontrollitav tekst*"

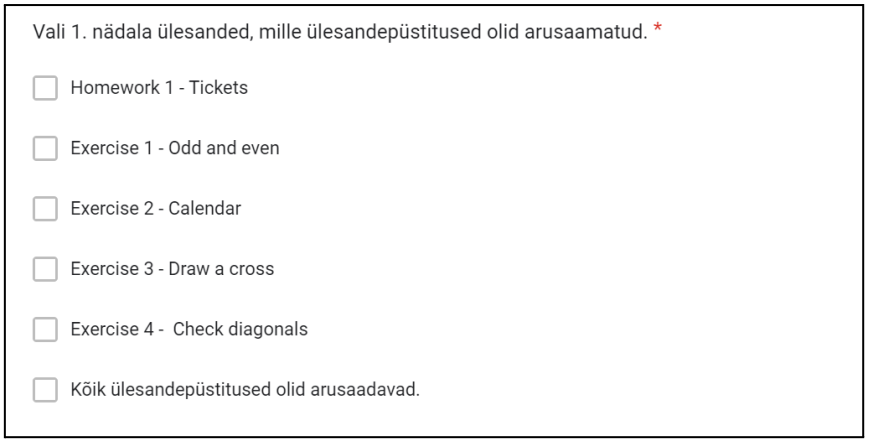
koosneb üheksast hinnatavast aspektist: sisu kvaliteet, õpieesmärkide kooskõla, tagasiside ja kohandamine, motiveerimine, esitluse kujundus, interaktsiooni kasutatavus, ligipääsetavus, taaskasutatavus ja standarditele vastavus [29-30]. Küsimustiku teel saadud tagasiside põhjal hinnati mudeli järgmisi aspekte:

- Sisu kvaliteeti (ingl *Content quality*)
- Õpieesmärkide kooskõla (ingl *Learning goal alignment*)
- Motiveerimist (ingl *Motivation*)
- Esitluse kujundus (ingl *Presentation Design*)
- Standarditele vastavus (ingl *Standards compliance*)

Lõputöö käigus uuendati 20 praktikumi ülesannet ja 9 kodust ülesannet, moodustades kokku 29 ülesannet, mis katab viie nädala jagu ülesandeid. Lisaks loodi igale kodutööle vastav murelahendaja, kokku 9 murelahendajat.

4.2 Ülesandekogule tagasiside küsimine

Käesolevas bakalaureusetöös kasutati loodud õppematerjalide hindamiseks tagasiside küsimustikku. Küsimustik saadeti kursuse "Introduction to programming II" praktikumi juhendajatele, et koguda tagasisidet loodud ülesandekogu kohta. Eesmärk oli välja selgitada, millised ülesanded olid liiga keerulised, liiga kerged, polnud asjakohased, olid arusaamatud või ei keskendunud piisavalt nädala põhiteemale. Tagasisidet koguti perioodil 3. märts kuni 17. märts 2023. Kuna õppematerjalid ei olnud kursuse toimumise hetkel veel valmis polnud võimalik tagasisidet saada üliõpilastelt. Pärast praktikumi juhendajate tagasisidet täiendati või vahetati välja ülesanded, mille kohta esines kriitikat. Murelahendajatele küsiti tagasisidet eraldi, pärast ülesandekogu täieliku valmimist, et saada sisukamat ja kvaliteetsemat lõpptulemust. Murelahendajate tagasiside küsimise protsessi kirjeldatakse detailsemalt peatükis 4.5.

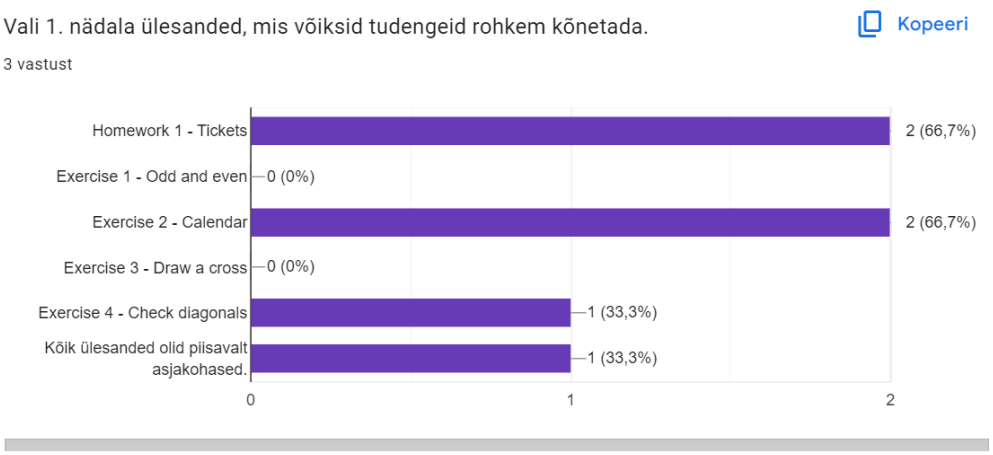


Joonis 7. Ülesandekogu esimese nädala ülesannete tagasiside küsimuse näide

Joonisel 7 on kujutatud tagasiside küsimus esimese nädala ülesannete kohta, mille eesmärk oli hinnata ülesannete tekstide arusaadavust. Iga nädala ülesannete kohta küsiti tagasisidet eraldi, valikvastusteks olid igal nädalal olevad praktikumiülesanded ja kodused ülesanded. Tagasiside küsimiseks kasutati keskkonda Google Forms, kuhu koostati küsimustik. Küsimustik koosnes 25 küsimusest (nii avatud kui suletud küsimused) ning vastajatel oli võimalus lisada iga nädala ülesannete kohta täiendavaid kommentaare (vt lisa 4), mida praktikumijuhendajad kasutasid aktiivselt. Kuigi kursus ja loodud materjalid olid inglise keeles, koostati küsimustik eesti keeles, kuna praktikumi juhendajad valdasid eesti keelt.

4.3 Ülesandekogu tagasiside analüüs

Tagasiside küsimustikule vastas kolm praktikumi juhendajat viiest. Saadud tagasiside oli piisav, et ülesandekogu täiendada ja parandada materjalide kvaliteeti. Igas nädalas esines üks või kaks ülesannet, mis vajasid täiendamist. Paranduste põhjused varieerusid sõltuvalt konkreetsest ülesandest, kuid üldiselt polnud ülesannete keerukus probleemiks. Enam levinud probleemideks olid ülesannete kõnetavus ehk ülesande tekstid ei olnud piisavalt asjakohased, ning ülesannete tekstide sõnastus ehk sisu arusaadavus. Esines ka ülesandeid, mis polnud piisavalt teemakohased ehk ei käsitlenud piisavalt vastava nädala temaatikat.



Joonis 8. Ülesandekogu tagasiside küsimuse vastus

Joonisel 8 on esitatud tagasiside küsimustiku tulemused seoses esimese nädala ülesannetega, mis ei olnud piisavalt asjakohased. Jooniselt on näha, et kahe ülesande puhul oli vaja teha täiendavaid parandusi ning samuti vaadati üle kolmas ära märgitud ülesanne.

Kommentaaride jätmine oli vabatahtlik, ometi jäeti iga nädala kohta mitmeid sisukaid kommentaare, tuues välja täiendamist vajavad aspektid ning pakuti välja potentsiaalsed lahendused. Üheks selliseks kommentaariks oli soovitus kasutada teistsugust terminit ühe ülesande sõnastuses ja seda vastavalt põhjendatud.

„*Exercise 1 - Odd and even: võiks kasutada läbivalt kas list või array mõistet. Kursusel on minu meelest kasutusel "list"*" (Praktikumijuhendaja 1)

„*Exercise 5 - Team captains: võib-olla mainida ka ära, et numbrite järjekord isikukoodis pole oluline.*" (Praktikumijuhendaja 2)

„*Excercise 1 juures muudaks seda, et funktsiooni defineerimine on enne põhiprogrammi osa. Näited võiks rohkem olla funktsiooni kohta erinevate 2x järjenditega. Samuti võiksid näited olla arvutuste juures.*" (Praktikumijuhendaja 3)

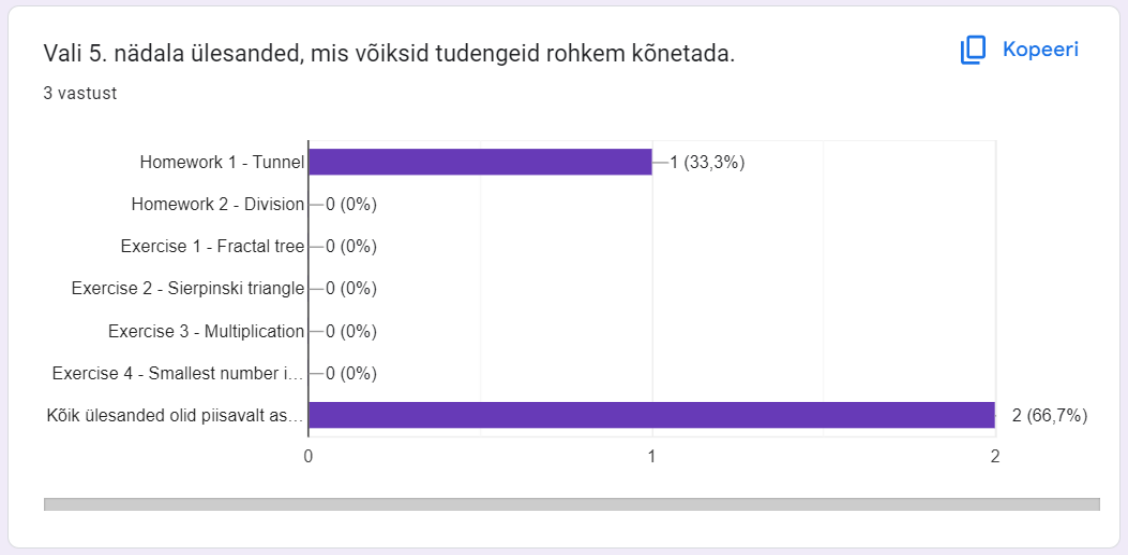
Kokku jäeti 10 kommentaari, millest kõiki võeti arvesse ja tehti vastavad muudatused ülesandekogus. Tagasiside jagunes kolmeks osaks: esiteks anti nõu sisuliste muudatuste tegemiseks, teiseks märgiti mõned teksti sõnastuse vead ning kolmandaks kiideti positiivseid ja toredaid ülesandeid.

Ülesanded, mille kohta jäeti kommentaare, olid järgmised:

- 1. nädala esimene ja teine praktikumi ülesanne
- 2. nädala teine kodune ülesanne ja teine praktikumi ülesanne
- 3. nädala esimene, neljas, viies ja kuues praktikumi ülesanne
- 5. nädala esimene praktikumi ülesanne

Lisaks jäeti kommentaarides ka üldisemaid soovitusi nagu rohkemate näidete lisamist ülesannete tekstidesse, et suurendada arusaadavust.

Kõiki kommentaaride soovitusi võeti arvesse, kuid mitte kõiki täienduste soovitusi valikvastustes ei viidud ellu, kuna lõputöö autor pidas mõnda neist ebavajalikuks või ei nõustunud nendega. Näiteks leidis üks praktikumi juhendaja, et viienda nädala esimene kodune ülesanne polnud piisavalt asjakohane ehk ei kõnetanud üliõpilast piisavalt (joonis 9).



Joonis 9. Tagasiside küsimustiku küsimuse tulemus

Seda arvamust ei võetud arvesse, kuna vaid üks vastaja kolmest leidis, et ülesanne võiks tudengeid rohkem kõnetada. Kõnetavus on subjektiivne ning lõputöö autor nõustus seekord kahe teise vastajaga, kes leidsid, et see ülesanne oli piisavalt asjakohane.

4.4 Murelahendajate loomine

Käesolevas lõputöös koostati murelahendajad kolmes etapis: esmalt tuvastati igas ülesandes võimalikud murekohad, seejärel loodi murekohtadele vastavad küsimused ning viimaks lisati küsimustele vastavad suunised. Kokku valmis üheksa murelahendajat (vt lisa 3), mis katab viie nädala koduülesanded. Igas nädalas on üks kuni kaks koduülesannet. Murelahendajad

loodi vaid kodutöödele, kuna praktikumides lahendatakse ülesandeid koos õppejõuga ning üliõpilased saavad abi saamiseks pöörduda praktikumi juhendaja poole. Sellest tulenevalt puudus vajadus praktikumi ülesannetele vastavate murelahendajate jaoks.

Murelahendajate loomine algas Tartu Ülikoolis varasemalt loodud murelahendajatega tutvumisega, sealhulgas eelmisel aastal loodud murelahendajatega kursuse „Introduction to programming" esimese osa jaoks. Seejärel analüüsiti iga loodud koduse ülesande võimaliku lahenduskäiku ja prooviti tuvastada kohad, mis võiksid üliõpilastele raskusi valmistada. See etapp on murelahendajate loomisel kõige keerulisem, kuna ülesande autor ei pruugi märgata mõnda näiliselt kerget kohta, mis lihtsusele vaatamata võib algajale raskusi valmistada. Murekohtade ülesmärkimiseks kasutati Google Docsi dokumenti. Pärast murekohtade kaardistamist sisestati abimaterjalid küsimuste ja vihjete komplektina keskkonda https://progtugi.cs.ut.ee/, kus on saadaval ka teised Tartu Ülikoolis kasutusel olevad murelahendajaid. Oluline küsimuste esitamisel oli, et üliõpilane tunneks ära, kui küsimus on seotud tema murega. Vihjete andmisel on tähtis, et vastust ei antaks ette, vaid vihjed suunaksid üliõpilase õigele teele. Vastuse ette andmine võib vähendada üliõpilaste iseseisvat materjali läbitöötamist ning suunata neid kohe murelahendajast lahendust otsima. Loodud murelahendajate tõhususe tagamiseks paluti praktikumijuhendajatelt murelahendajate kohta tagasisidet anda, vastavalt millele tehti neis täiendusi. Murelahendajate loomine algas pärast ülesandekogu valmimist ning valmisid 2023. aasta aprilli jooksul.

Oluline on siinkohal mainida, et sarnaselt ülesandekogu loomisele kasutati tehisintellekti ChatGPT[5] abi murelahendajate keelelisel toimetamisel selleks, et tekstidest eemaldada grammatika- ja kirjavead ning parandada lausete sõnastust.

Järgnevalt on kirjeldatud viienda nädala esimese kodutöö murelahendaja koostamise protsessi. Joonis 10 kujutab viienda nädala esimese kodutöö teksti, millele loodi vastav murelahendaja.

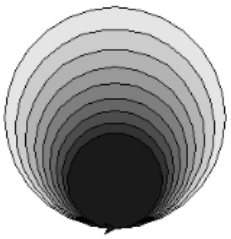
[5] Lõputöö autor on õigekirja ja grammatika kontrollimiseks ning teksti loetavuse parandamiseks rakendanud lõputöös, loodud ülesandekogus ja murelahendajates ChatGPT abi (2023), s.o kasutades keelemudelit, mille väljaõpe põhineb suurel hulgal erinevatel tekstiallikatel. ChatGPT on välja töötatud OpenAI poolt. Lisateavet ChatGPT ja OpenAI kohta: https://openai.com. Mudelile esitatud sisend: "Kontrolli õigekirja ja grammatikat: *kontrollitav tekst*"

Write a function that takes the depth, radius and color as parameters and draws a tunnel using turtle. With each iteration, the tunnel has to get darker in color. Use the color 'gray' variations (*gray0, gray10, gray20 ...*) from this website: https://cs111.wellesley.edu/labs/lab02/colors.

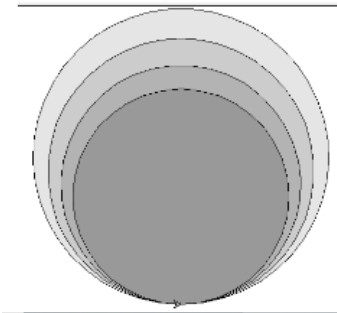
Hint:

- Use **t.fillcolor("black")** to adjust color.
- Use **t.begin_fill()** and **t.end_fill()** to fill an object with color.

Example tunnel with depth 10 and radius 100:



Example tunnel with depth 5 and radius 200:



Joonis 10. 5. nädala 1. kodutöö

Iga ülesande puhul alustati murelahendaja loomist ülesandele vastava näidislahenduse kirjutamisega. Joonisel 11 on näha selle konkreetse ülesande näidislahendus, mida jagati vaid praktikumi juhendajatega, et nad saaksid murelahendajaid efektiivsemalt ning sisukamalt hinnata.

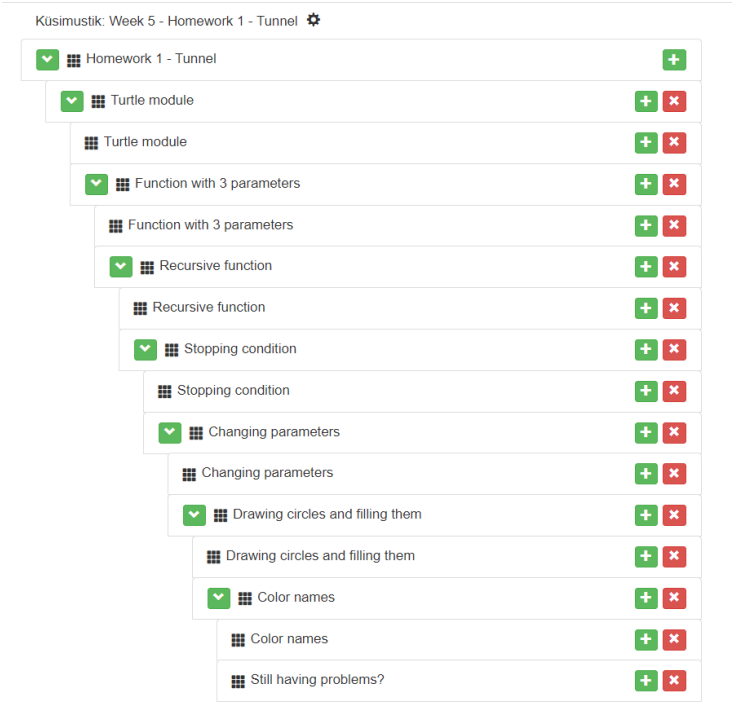
```
import turtle

t = turtle.Turtle()
t.speed(10)
def tunnel(d, r, g=90):
    if d <= 1: return
    t.fillcolor("gray" + str(g))
    t.begin_fill()
    t.circle(r)
    t.end_fill()
    tunnel(d-1, r * 0.9, g-10)

tunnel(10, 100.0)
```

Joonis 11. 5.nädala 1. kodutöö näidislahendus

Järgmisena jagati näidislahendus etappideks, et iga etapi jaoks koostada vastav küsimus. Küsimused koostati algselt Google Docsi dokumendis, et neid oleks vajadusel lihtne ja mugav muuta. Vaadeldav ülesanne jagunes järgnevateks küsimusteks:

- Kas üliõpilane oskab importida Turtle moodulit ning seda ka kasutada?
- Kas üliõpilane oskab luua funktsiooni kolme parameetriga?
- Kas üliõpilane oskab luua rekursiivset funktsiooni, millel on peatumis tingimus?
- Kas üliõpilane oskab ringi joonistada ja seda värviga täita?

Nendele küsimustele tuginedes loodi progtugi keskkonda ülesandele vastav murelahendaja, mida on kujutatud joonisel 12.



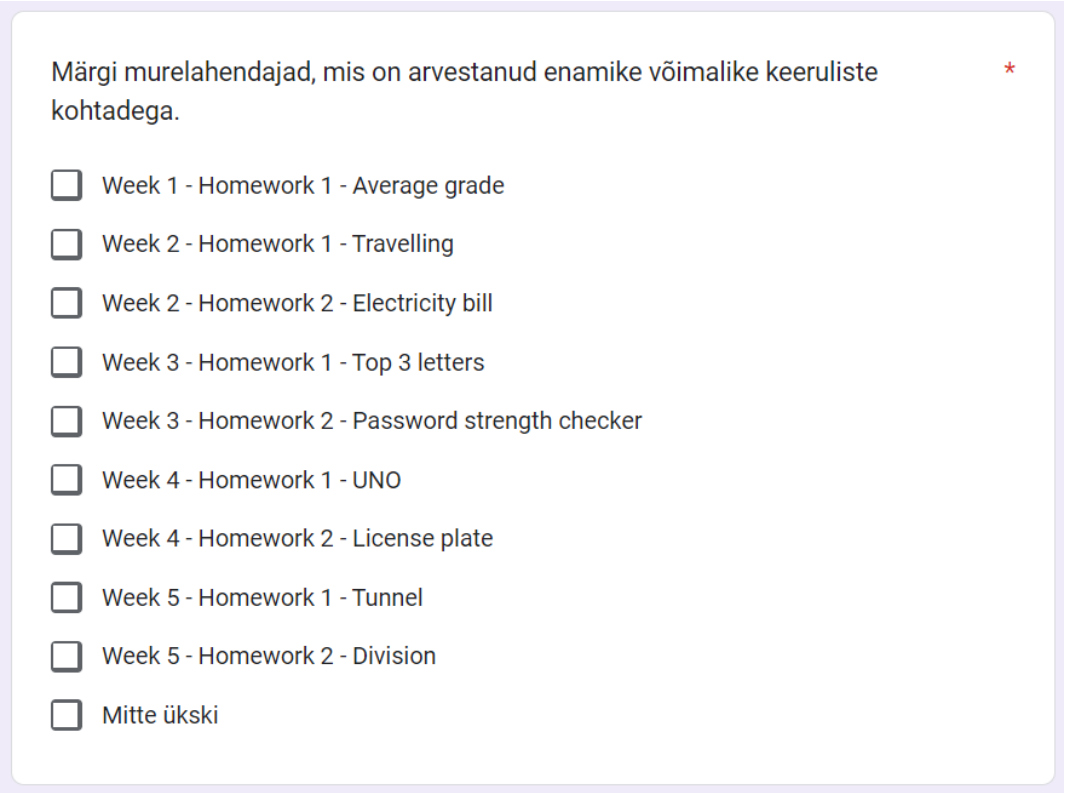
Joonis 12. 5.nädala 1. kodutöö murelahendaja

Joonisel 12 on viienda nädala esimese kodutöö murelahendaja. Pildilt on näha murelahendaja struktuuri, kus küsimusi esitatakse ülevalt alla. Kui kasutaja enda probleemile lahendust ei saanud, siis soovitatakse tal pöörduda praktikumi juhendaja poole.

4.5 Murelahendajatele tagasiside küsimine

Sarnaselt ülesandekogule küsiti tagasisidet murelahendajatele (vt lisa 5) vaid kursuse praktikumi juhendajatelt, kuna õppematerjalid ei olnud kursuse toimumise hetkel veel valmis, et üliõpilastelt tagasisidet saada. Tagasiside küsitluse koostamisele eelnes murelahendajate loomine https://progtugi.cs.ut.ee/ keskkonda. Tagasiside saamiseks kasutati Google Formsi keskkonda, kuhu koostati küsimustik. Tagasisidet koguti perioodil 3. aprill kuni 17. aprill

2023. Küsimustiku (vt lisa 5) eesmärgiks oli hinnata, kas murelahendajates on keeruliste kohtadega arvestatud ning kas murelahendajad aitavad nendele murekohtadele lahendust leida, sealjuures vastust ette andmata. Samuti sai iga murelahendaja kohta jätta täiendavaid kommentaare.

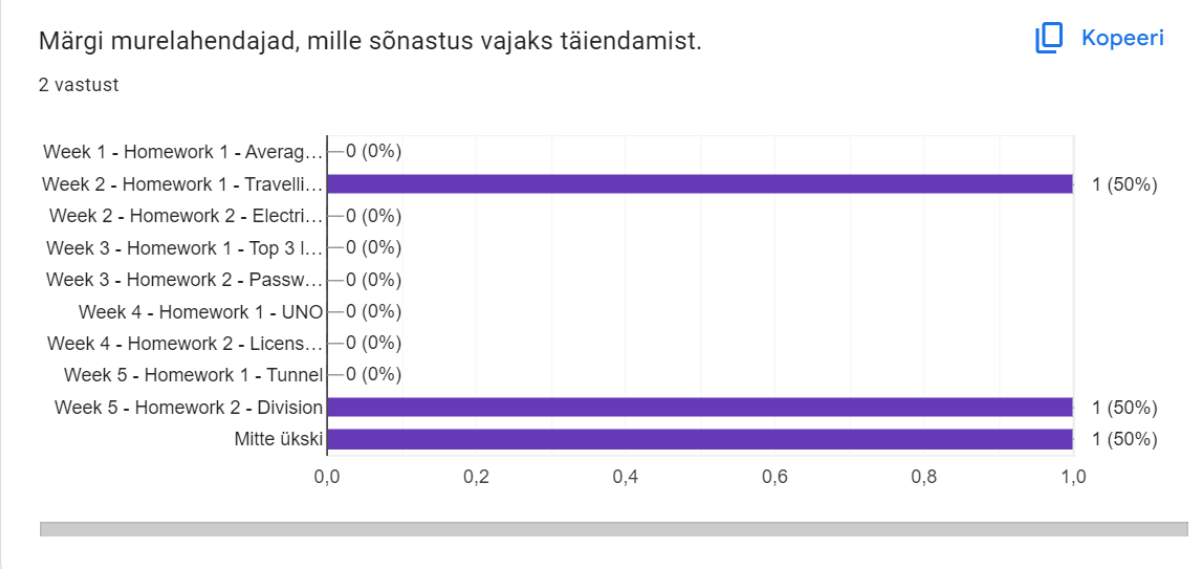


Joonis 13. Murelahendajate tagasiside küsimuse näide

Joonisel 13 on kujutatud tagasiside küsimustiku küsimust, mille eesmärgiks oli saada hinnangut selle kohta, kas loodud murelahendajad arvestavad võimalikult paljude keeruliste kohtadega. Valikvastuste variantideks olid kõikide kodutööde pealkirjad. Küsimustik koosnes kokku neljast küsimusest (avatud küsimused) ning vastajatel oli võimalus lisada iga murelahendaja kohta täiendavaid kommentaare.

4.6 Murelahendajate tagasiside analüüs

Murelahendajate tagasiside küsimustikule (vt lisa 5) vastas kaks praktikumi juhendajat viiest. Saadud tagasiside oli piisav, et murelahendajaid täiendada ning paremaks teha. Üldiselt vastati üksmeelselt, et murelahendajad arvestavad kõikide võimalike murekohtadega, aitavad tekkinud probleeme lahendada ning ei anna seejuures lahendust ette. Kaks murelahendajat märgiti, mille sõnastust tuleks täiendada (vt joonis 14).



Joonis 14. Murelahendajate küsimustiku küsimuse tulemused

Kommentaaride jätmine oli taaskord vabatahtlik, kuid taaskord jäeti mitu sisukat kommentaari ja soovitust. Kokku jäeti 3 kommentaari ning neid kõiki võeti arvesse. Üheks kommentaariks oli soovitus lisada täiendavaid selgitusi teise nädala esimese kodutöö murelahendajale. Murelahendajat täiendati vastavalt, lisades selgitusi selle kohta, miks ülesandes tuleb teha *split* tegevust.

„*"Does your program split the date input into an array?" - tudengile äkki ei ole selge, miks vaja split teha on. Võib-olla seda lauset täiendada?*". (Praktikumijuhendaja A)

„*"Does your program convert the input to lowercase?" -> Does your program convert the input to lowercase as upper- and lowercase characters are consider otherwise different?*". (Praktikumijuhendaja A)

„*Äkki lisada ka teksti joonistamise näide?*". (Praktikumijuhendaja B)

Lõputöö praktilises osas uuendati kursuse „Introduction to programming II" ülesandekogu, luues 20 uut praktikumi ülesannet ja 9 kodust ülesannet. Ülesandekogu loomisel rakendati ADDIE mudelit ning ülesannete kohta küsiti tagasisidet praktikumi juhendajatelt. Lisaks loodi kodutöödele vastavad murelahendajad, millele küsiti tagasisidet praktikumi juhendajatelt. Saadud tagasiside analüüsiti ning õppematerjale täiendati vastavalt soovitustele. Järgmises peatükis (vt ptk 5) on analüüsitud loodud õppematerjale ja võrreldud teiste sarnaste kursuste materjalidega.

5. Loodud õppematerjalide analüüs

Õppematerjalide koostamisel oli oluline tagada, et materjalide sisu ja teemakäsitlus jääksid samaks. Ülesannete loomisel oli tähtis, et loodavate tekstide sisud on arusaadavad, keerukuselt sobivad ja kõnetaksid üliõpilasi. Selleks küsiti materjalidele tagasisidet kursuse praktikumi juhendajatelt, kelle hinnangul loodud materjalid vastasid tingimustele. Ülesanded, mis vajasid nende arvates täiendamist, täiendati vastavalt. Samuti saadi tagasisidet murelahendajate arusaadavuse ja efektiivsuse kohta. Lisaks tagasisidele kasutati Grammarly ja ChatGPT abi, selleks, et tekstidest eemaldada kirja- ja grammatikavead ning parandada lausete sõnastust.

Kursusel „Introduction to programming II" kasutatakse ümberpööratud klassiruumi meetodit, mistõttu loodi ülesanded, arvestades selle meetodiga. Meetodi kohaselt on oluline roll iseseisval õppimisel ning teema kinnistatakse praktikumides. Üliõpilased tutvuvad teemaga iseseisvalt läbi loengumaterjalide ja teoreetilise osa ning kodutööd on natuke keerulisemad, et kontrollida õpilaste teemast arusaamist. Praktikumid toimuvad iga teema lõpus, pärast teiste materjalidega töötamist ning sisaldavad kodutööst veelgi raskemaid ülesandeid, et suurendada üliõpilaste teadmisi. Selline lähenemine ülesandekogule oli ka varasemalt kursusel kasutusel.

Esimeses peatükis (vt ptk 1.3) käsitletakse kursusel varasemalt kasutusel olnud ülesannete tüüpe, kus on välja toodud, et koduste ja praktikumiülesannete hulgas kasutati ülesandeid, mis kuulusid gruppi "Rakenda ja analüüsi". Ülesanded jagunesid omakorda kahte alamkategooriasse "Teosta" ja "Kohanda". Käesoleva lõputöö raames on loodud ülesanded, kus üliõpilastel tuleb samuti rakendada olemasolevaid teadmisi, et keerukamate ülesannete puhul koodist aru saada, seda ise kirjutada või modifitseerida. Seega on uue ülesandekogu ülesannete tüübid jäänud samaks.

Õppematerjalide võrdluseks on kasutatud teisi Tartu Ülikooli programmeerimise algkursusi nagu "Programmeerimine", "Programmeerimisest maalähedaselt" ja "Introduction to programming" esimene osa. Kuigi kursuste maht ja ülesannete arv on neil erinev, on kõikidel kursustel ülesanded samal või sarnastel teemadel ning ülesannete tekstidesse on lisatud elulisi aspekte, et suurendada üliõpilaste motivatsiooni lahendamisel. Kõikidel kursustel kasutatakse ka murelahendajaid, mis paiknevad samuti keskkonnas www.progtugi.cs.ut.ee. Igal kursusel

on murelahendaja eesmärk sama, mis on üliõpilaste toetamine ülesannete lahendamisel, ilma vastuseid ette andmata. Enamik kursustest on eestikeelsed, seega on murelahendajad samuti eesti keeles. Kursus "Introduction to programming" oli esimene, millele loodi murelahendajad inglise keeles, kuna kursus ise on samuti ingliskeelne. Selle kursuse murelahendajatest on võetud inspiratsiooni lahendamaks probleeme, mis tulenevad sellest, et murelahendajate keskkond on eestikeelne. Üheks selliseks probleemiks oli murelahendajate keskkonnas olev tagasi liikumise nupp, mis on eestikeelne ning mida pole võimalik muuta.

Ühe võimaliku edasiarendusena võiks selles töös olla juturoboti loomine murelahendajate asemel, kasutades tehisintellekti, kuid piirates roboti võimalust vastust ette anda. Selline juturobot tagaks, et ülesande lahendajad ei jääks mõne arvestamata murekoha tõttu hätta. Samas võiks murelahendaja eeliseks olla olukord, kus üliõpilane ei tea täpselt, mis tema lahenduses puudu on ning ta avastab selle läbi murelahendaja esitatud küsimuste.

6. Kokkuvõte

Käesoleva bakalaureusetöö eesmärk oli täiendada Tartu Ülikooli programmeerimise algkursuse "Introduction to programming II" õppematerjale. Täpsemalt loodi uus ülesandekogu, 20 praktikumi ülesannet ja 9 kodutööd. Lisaks koostati igale kodutööle vastav murelahendaja, mis toetavad üliõpilasi ülesannete lahendamisel. Loodud õppematerjalidega loodetakse tõsta üliõpilaste motiveeritust ja vähendada õppejõudude koormust. Ülesandekogu loomisel rakendati ADDIE õppematerjalide loomise mudelit. Murelahendajate loomisel tutvuti varasemalt loodud murelahendajatega ning lähtuti autori oskusest tuvastada peamised murekohad. Kursus "Introduction to programming II" on ingliskeelne, seega on ka loodud õppematerjalid inglise keeles. Loodud õppematerjalide kvaliteedi tagamiseks kasutati rakendust Grammarly ning tehisintellekti ChatGPT, et ülesannete sõnastused oleks nii grammatiliselt korrektsed kui ka kirjavigade vabad. Samuti koguti lõputöös tagasisidet ülesandekogule ja murelahendajatele kursuse praktikumi juhendajatelt, kasutades selleks Google Formsi loodud küsimustikke. Tagasisidele tuginedes täiendati loodud õppematerjale, et suurendada loodud materjalide kvaliteeti.

Lõputöö teoreetilises osas kirjeldatakse kursust, ADDIE mudelit ja tutvustatakse murelahendajaid. Pärast teoreetilist osa tutvustatakse lõputöö praktilist osa, kus kirjeldatakse loodud õppematerjalide koostamise protsessi. Praktilise osa tulemusena loodi kokku 29 ülesannet ning 9 murelahendajat. Praktilises osas on lisaks antud ülevaade tagasiside kogumise protsessist ja analüüsist.

Autori loodud õppematerjale rakendatakse "Introduction to programming II" kursusel 2023. aasta sügissemestril. Pärast seda kursust on oluline koguda tagasisidet üliõpilastelt, et saada põhjalik ülevaade õppematerjalide tugevatest ja nõrkadest külgedest ning teha vajadusel täiendusi.

Viidatud kirjandus

[1] Tartu Ülikooli kursus „Introduction to programming II”.

https://courses.cs.ut.ee/2021/itpII (06.02.2023)

[2] Tartu Ülikooli õppeinfosüsteem. https://ois2.ut.ee/ (02.03.2023)

[3] Lepp, M., Palts, T., Luik, P., Papli, K., Suviste, R., Säde, M., Hollo, K., Vaherpuu, V., Tõnisson, E. Troubleshooters for Tasks of Introductory Programming MOOCs.

International Review of Research in Open and Distributed Learning, Vol. 19, No. 4, 2018.

https://www.erudit.org/en/journals/irrodl/2018-v19-n4-irrodl04233/1055528ar/ (10.01.2023)

[4] TIOBE programmeerimiskeelte indeks. https://www.tiobe.com/tiobe-index/ (01.05.2023)

[5] Bogdanchikov, A., Zhaparov, M., Suliyev, R. Python to learn programming. *Journal of Physics Conference Series*, 2013.

https://www.researchgate.net/publication/258800484_Python_to_learn_programming (10.01.2023)

[6] Tartu Ülikooli kursus „Programmeerimine”. https://courses.cs.ut.ee/2023/progeng/spring (06.02.2023)

[7] Aşıksoy, G., Ozdamli, F. Flipped Classroom Approach. *World Journal on Educational Technology Current Issues*, 2016.

https://www.researchgate.net/publication/309890120_Flipped_Classroom_Approach (15.03.2023)

[8] Taşpolat, A., Ozdamli, F., Soykan, E. Programming Language Training With the Flipped Classroom Model. *SAGE Open*, 2021, pp. 1–17.

https://www.researchgate.net/publication/352318196_Programming_Language_Training_With_the_Flipped_Classroom_Model/link/643c0cc5a08d9a67a4a024e2/download (15.03.2023)

[9] Díaz, M.J.S., Antequera, J.G., Pizarro, M.C. Flipped Classroom in the Context of Higher Education: Learning, Satisfaction and Interaction. 2021.

https://www.mdpi.com/2227-7102/11/8/416 (15.03.2023)

[10] Severance C. Veebikursus "Python for Everybody". https://www.py4e.com (02.03.2023)

[11] Kumar, A. Learning Programming by Solving Problems. *IFIP Advances in Information and Communication Technology*, 2002, pp. 29-39.

https://www.researchgate.net/publication/221545525_Learning_Programming_by_Solving_Problems (20.01.2023)

[12] Nwosisi, C., Ferreira, A., Rosenberg, W., Walsh, K. A Study of the Flipped Classroom and Its Effectiveness in Flipping Thirty Percent of the Course Content. *International Journal of Information and Education Technology*, 2016, pp. 348-351. https://www.researchgate.net/publication/276427778_A_Study_of_the_Flipped_Classroom_and_Its_Effectiveness_in_Flipping_Thirty_Percent_of_the_Course_Content (20.01.2023)

[13] Amresh, A., Carberry, A. R., Femiani, J. Evaluating the effectiveness of flipped classrooms for teaching CS1. *IEEE Frontiers in Education Conference (FIE)*, 2013. https://ieeexplore.ieee.org/abstract/document/6684923?casa_token=0_8hCGa_ryIAAAAA:SFAABxJMXnMEWd7K2RghldQSyVTmmaPk5FU9un10bZ1LA1k5Si2sPSJeh_48fCQm-fW9TVto-w (20.01.2023)

[14] Veebileht ,,Programmeerimise ülesannete tüübid'' https://courses.cs.ut.ee/t/progylTyybid (03.05.2023)

[15] Schihalejevi, R. Õppeprotsessi kognitiivsetele tasemetele vastavad ülesanded gümnaasiumi kursusel „Programmeerimine". TÜ arvutiteaduste instituudi magistritöö. 2021. https://dspace.ut.ee/bitstream/handle/10062/72265/schihalejev_ruth_ma_2021.pdf?sequence=1&isAllowed=y (03.05.2023)

[16] Seemiller, C., Grace, M., Campagnolo, P. D. B., Alves, I. M. D. R., Borba, G. S. D. What makes learning enjoyable? Perspectives of today's college students in the U.S. and Brazil. *Journal of Pedagogical Research,* Vol. 5, Issue 1, 2021. https://files.eric.ed.gov/fulltext/EJ1289019.pdf (20.01.2023)

[17] Saarevet, T. Koduülesannete ja murelahendajate loomine kursusele „Introduction to Programming". TÜ arvutiteaduste instituudi bakalaureusetöö. 2022. https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=74534&year=2022 (10.04.2023)

[18] Jonassen, D.H., Hung, W. Learning to Troubleshoot: A New Theory-Based Design Architecture. *Educ Psychol Rev* 18, 2006, pp. 77–114. https://doi.org/10.1007/s10648-006-9001-8 (20.04.2023)

[19] Tartu Ülikooli Murelahendajate veebileht. https://progtugi.cs.ut.ee/ (10.04.2023)

[20] Vaherpuu, V.. Murelahendajate loomise keskkond. TÜ arvutiteaduste instituudi bakalaureusetöö. 2016. https://dspace.ut.ee/handle/10062/56219 (10.04.2023)

[21] Tartu Ülikooli Informaatika Didaktika töörühma veebileht. https://didaktika.cs.ut.ee/tehtud-tood/ (10.04.2023)

[22] Tartu Ülikooli kursus ,,Programmeerimisest maalähedaselt''. https://courses.cs.ut.ee/2023/progmaa/spring (10.04.2023)

[23] Kaimre, J. Murelahendajate koostamine Tartu Ülikooli kursuse „Programmeerimine“ jaoks. TÜ arvutiteaduste instituudi bakalaureusetöö. 2021. https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=71614&year=2021 (10.04.2023)

[24] Klaanberg, A. Murelahendajate koostamine Tartu Ülikooli kursuse „Objektorienteeritud programmeerimine” tarbeks . TÜ arvutiteaduste instituudi bakalaureusetöö. 2020. https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=69730&year=2020 (10.04.2023)

[25] Bouchrika, I. Instructional Design Models: ADDIE, Gagne’s, Merrill’s and Bloom’s Methodologies. https://research.com/education/instructional-design-models (29.11.2022)

[26] Bouchrika, I. The ADDIE Model Explained: Evolution, Steps, and Applications. https://research.com/education/the-addie-model (04.05.2023)

[27] Aldoobie, N. ADDIE Model. *American International Journal of Contemporary Research* Vol. 5, No. 6, 2015. http://www.aijcrnet.com/journals/Vol_5_No_6_December_2015/10.pdf (23.02.2023)

[28] Downes, A. Waltershed veebileht. ADDIE: A 5-Step Process for Effective Training & Learning Evaluation. https://www.watershedlrs.com/blog/learning-evaluation/addie-instructional-design-model/ (05.05.2023)

[29] Leacock, T. L., Nesbit, J. C. A Framework for Evaluating the Quality of Multimedia Learning Resources. *Educational Technology & Society*, 2007, pp. 44-59. http://www.sfu.ca/~jcnesbit/articles/LeacockNesbit2007.pdf (29.04.2023)

[30] Nesbit, J., Belfer, K., Leacock, T. Learning Object Review Instrument (LORI 1.5). User Manual. 2004. http://web.archive.org/web/20040126041853/http://elera.matchbox.surrey.sfu.ca/eLera/Home/Articles/LORI%201.5.pdf (05.05.2023)

Lisad

I. Tehisintellekti kasutamise näide

kontrolli õigekirja ja grammatikat: Lõputöö viimane sisupeatükk on kokkuvõte, millele järgneb lisade peatükk, mis sisaldab loodud ülesandekogu, murelahendajaid ning tagasisideküsimustike.

Lõputöö viimane sisupeatükk on kokkuvõte, millele järgneb lisapeatükk, mis sisaldab loodud ülesandekogu, murelahendajaid ning tagasisideküsimustikke.

II. Koostatud ülesandekogu

Nädal 1. Kodutöö 1 (Homework 1 - Average grade)

The file "grades.txt" contains all of your grades for each subject. Each line corresponds to one subject and contains multiple grades separated by a space. The grades are in letter form (A, B, C, D, E, F). To calculate the average grade, they must first be converted to numeric values.

Conversion:

- A --> 5
- B --> 4
- C --> 3
- D --> 2
- E --> 1
- F --> 0

Write a function called *averageGrade()* that takes a filename as an argument. The function should read the file into a list of lists. Then the program should iterate through that list and find the average **final** grade. Final grades are calculated by summing all the grades for a subject and dividing by the count of them. Round the answer to 2 decimal points.

Example grade.txt file:

```
A B B B D A
C C C D E B
```

```
A A B
B A B B C
C C B E
F A A B A
```

Output of the example:

```
Your average grade is 3.65
```

Nädal 1. Praktikumi ülesanne 1 (Exercise 1 - Odd and even)

Write a program that reads a text file containing a matrix of numbers. Each line in the file represents a row of the matrix, and each number is separated by a space.

The program should parse the file and create a 2D list where each sublist is a row from the file.

Write a function that takes this list of lists as an argument and calculates the result of the following equation:

- The total value at the beginning is 1.
- For each row in the matrix:
    1. If the row index is even:
        - Add up all the even-indexed elements in the row to the row total.
        - Subtract all the odd-indexed elements in the row to the row total.
        - Multiply the total value by the row total value.
    2. If the row index is odd:
        - Add up all the even-indexed elements in the row to the row total
        - Subtract all the odd-indexed elements in the row to the row total.
        - Divide the total value by the row total value.

The function should return the final result of the calculation.

Example 1. Example txt file:

```
1 2 3
2 2 1
3 2 3
```

Example calculation for this txt file:

```
total = 1
total = total * (1-2+3)
total = total / (2-2+1)
total = total * (3-2+3)
```

Output of the example:

```
8.0
```

Example 2. Example txt file:

```
0 2 3 1
1 0 1 1
2 3 1
```

Example calculation for this txt file:

```
total = 1
total = total * (0-2+3-1)
total = total / (1-0+1-1)
total = total * (2-3+1)
```

Output of the example:

```
0.0
```

Nädal 1. Praktikumi ülesanne 2 (Exercise 2 - Bus seat reservation system)

A newly founded bus transfer company that operates between Tallinn and Tartu has approached you to develop a seat reservation system for their buses.

All seats are labeled in the same way across all buses, with each seat label consisting of two parts: the row number and the column letter (A, B, C, D).
For example, seat number 2C is located in the second row and is the third seat from the left.

Write a function called *seating()* that takes the number of rows as a parameter and returns a

matrix that contains the seat labels for each row of the bus.

- Note that every row has 2 + 2 seats except for the last row, which has one extra seat in the middle (A, B, C, D, E).

After that, the program should use a while loop to prompt the user for a seat number. The program should then check the availability of the seat based on the previously created matrix, return the price of the ticket, and remove the seat from the matrix so that it is no longer displayed as available.

The price of the ticket depends on the seat location:

- Middle seats (B,C): 8 euros
- Window seats (A,D): 9 euros
- In the last row (B,C,D) are considered as middle seats and (A,E) as window seats.

Example:

```
>>>seating(5)
[['1A', '1B', '1C', '1D'], ['2A', '2B', '2C', '2D'], ['3A', '3B', '3C',
'3D'], ['4A', '4B', '4C', '4D'], ['5A', '5B', '5C', '5D', '5E']]
Please enter seat number (e.g. 1A): 3F
Sorry, that seat is not available.
Please enter seat number (e.g. 1A): 2A
Price of ticket: 9 euros
[['1A', '1B', '1C', '1D'], ['2B', '2C', '2D'], ['3A', '3B', '3C', '3D'],
['4A', '4B', '4C', '4D'], ['5A', '5B', '5C', '5D', '5E']]
```

Nädal 1. Praktikumi ülesanne 3 (Exercise 3 - Draw a cross)

Write a function called *cross()* that takes a positive number that is divisible by 3 as a parameter for the grid size and returns a matrix that contains a cross. The cross is symmetrical, and the width of the cross line is one-third of the matrix size.

Example 1 of a function call:

```
>>>cross(3)
[' ', 'X', ' ']
['X', 'X', 'X']
[' ', 'X', ' ']
```

Example 2 of a function call:

```
>>>cross(6)
[' ', ' ', 'X', 'X', ' ', ' ']
[' ', ' ', 'X', 'X', ' ', ' ']
['X', 'X', 'X', 'X', 'X', 'X']
['X', 'X', 'X', 'X', 'X', 'X']
[' ', ' ', 'X', 'X', ' ', ' ']
[' ', ' ', 'X', 'X', ' ', ' ']
```

Nädal 1. Praktikumi ülesanne 4 (Exercise 4 - Check diagonals)

Write a program that reads in a 2D list from a file "diagonals.txt". The file consists of integers. Check if each diagonal from left to right of that matrix has the same numbers throughout a diagonal.

Note: The height and width of the matrix are equal.

Example 1 of diagonals.txt:

```
1 2 6 1
2 1 2 6
3 2 1 2
4 3 2 1
```

Output of the example:

```
Correct!
```

Example 2 of diagonals.txt:

```
1 2 6 1
2 4 2 6
3 2 4 2
4 1 2 1
```

Output of the example:

```
Incorrect!
```

Nädal 2. Kodutöö 1 (Homework 1 - Travelling)

Write a program that prompts the user to enter countries they have visited and the travel start dates written in the format dd.mm.yyyy. The program should then create a dictionary where the keys are the names of the months and the values are the countries they visited that month. The user should be able to enter as many countries as they want. Once the user is done entering countries, the program should print out the dictionary and the month they have traveled to most frequently. Use *max()* function to find the country they have traveled most frequently.

Additionally, the program should have a function *month_name()* that takes the number of a month as its argument and returns the name of the month. The function should not contain if-statements.

Example:

```
Enter a country you have visited (or 'quit' to exit): Estonia
Enter the travel start date (in the format dd.mm.yyyy): 12.11.2021
Enter a country you have visited (or 'quit' to exit): Portugal
Enter the travel start date (in the format dd.mm.yyyy): 24.07.2022
Enter a country you have visited (or 'quit' to exit): Norway
Enter the travel start date (in the format dd.mm.yyyy): 10.07.2018
Enter a country you have visited (or 'quit' to exit): Denmark
Enter the travel start date (in the format dd.mm.yyyy): 15.02.2023
Enter a country you have visited (or 'quit' to exit): Sweden
Enter the travel start date (in the format dd.mm.yyyy): 24.02.2019
Enter a country you have visited (or 'quit' to exit): Turkey
Enter the travel start date (in the format dd.mm.yyyy): 12.07.2020
Enter a country you have visited (or 'quit' to exit): quit
```

Output:

```
Dictionary: {'November': ['Estonia'], 'July': ['Portugal', 'Norway',
'Turkey'], 'February': ['Denmark', 'Sweden']}
Most frequently traveled month:  July
```

Nädal 2. Kodutöö 2 (Homework 2 - Electricity bill)

Due to the recent increase in energy prices, you may be wondering which items in your home consume the most power.

Write a program that reads information about the energy consumption from a file, and then creates a dictionary where the key is an item (such as a laptop), and the value is the energy consumption in kilowatts (kW).

Each line in the file represents an energy consumer and contains the following information: the item name, the watts it uses per hour, and the number of hours it was used that month. Note that values in the file are separated by spaces, and items can occur multiple times.

To calculate the energy consumption:

- kW = W / 1000
- Monthly energy usage = hours * kW

After the dictionary is created, the program should output the total energy consumption in kilowatts and the item with the highest energy consumption.

NB! The use of *sum()* and *max()* functions is not allowed.

Example text file:

```
Lamp 60 60
Oven 2300 12
Washer 3500 10
Lamp 60 50
Laptop 50 70
ElectricHeater 1500 100
```

Output:

```
Total energy consumption: 222.7 kW.
Item with highest energy consumption: ElectricHeater.
```

Nädal 2. Praktikumi ülesanne 1 (Exercise 1 - Input to Dictionary of Letters)

Write a program that asks the user for input (a sentence) and converts that input to a dictionary of letters. The key in the dictionary should be the letter, and the value is the number of times that the letter occurs in the input. Ask the user for inputs until the user writes "end" as the input.

Don't put characters that are not letters into the dictionary—for example ',' and spaces. The case of the letters should not matter.

Output the dictionary created.

- **Hint 1:** Use character.isalpha() to check whether the character is a letter.
- **Hint 2:** Use string.lower() to make the word lowercase.

Example:

```
Print 'end' to stop.
Enter something: Hello, my name is Ben.
Enter something: end
{'h': 1, 'e': 3, 'l': 2, 'o': 1, 'm': 2, 'y': 1, 'n': 2, 'a': 1, 'i': 1,
's': 1, 'b': 1}
```

Nädal 2. Praktikumi ülesanne 2 (Exercise 2 - Average grade vol 2)

Modify the previous week's homework (Session 1 Homework 1) to use a dictionary instead of a 2D list. The file "grades.txt" contains all of your grades for each subject. Each line corresponds to one subject and contains multiple grades separated by a space. The grades are in letter form (A, B, C, D, E, F).

To calculate the average grade, they must first be converted to numeric values using the following scale:

- A --> 5
- B --> 4
- C --> 3
- D --> 2
- E --> 1
- F --> 0

Write a function called *averageGrade()* that takes a filename as an argument. The function should read the grades from the file and store them in a dictionary where the keys are the letter grades and the values are the counts of each letter grade. Then the program should iterate through that dictionary to find the average grade. Average grade is calculated by summing all the grades together and dividing by the count of them. Round the answer to 2 decimal points.

Example grade.txt file:

```
A B C B D
A C B D B
A C E
B A B C C
C C B E
F A F B C
```

Output of the example:

```
Your average grade is 3.22
```

Nädal 2. Praktikumi ülesanne 3 (Exercise 3 - Fridge & recipes)

Write a program that reads in a shopping bill from a text file and creates a dictionary where the key is the item, and the value is the amount of that item. Values(item and quantity) in a line are separated by a comma. Each line corresponds to one item. After that, the program should ask the user for an input of a recipe. In the input, the item has to be written as many times as it is needed.

Example text file:

```
apple,2
cream,1
milk,1
chocolate,1
carrot,3
```

The program has to check what items are missing. If nothing is missing, then notify the user that the fridge has all the recipe items.

Example 1:

```
Enter a recipe: apple,apple,apple,carrot,chocolate
The items you are missing are following: ['apple']
```

Example 2:

```
Enter a recipe: cream,apple,carrot,carrot,chocolate
You have all the items required for the recipe.
```

Nädal 3. Kodutöö 1 (Homework 1 - TOP 3 letters)

In this task, you will modify the "Input to Dictionary of Letters" program from the previous session (Session 2 Exercise 1) to find the three most common letters in a given set of inputs.

Write a program that prompts the user for multiple inputs, converts them into a dictionary of letters, and creates a list of tuples from the dictionary keys and values. The program will then sort the list in reverse order by the values and output the top 3 most frequently occurring letters.

Note: The dictionary key should be the letter and the value should be the count of that letter.

Example:

```
Print 'end' to stop.
Enter something: The weather is nice!
Enter something: It is really cold and rainy.
Enter something: end
{'t': 3, 'h': 2, 'e': 5, 'w': 1, 'a': 4, 'r': 3, 'i': 5, 's': 2, 'n': 3,
'c': 2, 'l': 3, 'y': 2, 'o': 1, 'd': 2}
```

Output:

```
i 5
e 5
a 4
```

Nädal 3. Kodutöö 2 (Homework 2 - Password strength checker)

Write a program that prompts the user for an email and a password, and then checks whether the password is strong enough.

A password is considered strong if it satisfies the following criteria:

- Password length >= 8 characters
- Unique characters in password >= 7

After checking the strength of the password, the program will check whether the password is too similar to the email. If the count of common letters between the email and password is more than half of the password length, the password will be considered too similar to the email.

Finally, the program should output whether the password is strong and different enough from the email.

Instructions:

- Define a function ***check_password_strength()*** that takes a string argument password and returns a boolean value indicating whether the password is strong or not.
- Define a function ***check_password_similarity()*** that takes two string arguments email and password and returns a boolean value indicating whether the password is too similar to the email or not.

Example 1:

```
Please enter your email: example@email.com
Please enter a password: $tr0ngP@ssword
Strong password and different enough from email!
```

Example 2:

```
Please enter your email: example@email.com
Please enter a password: password
Password is too weak. Please choose a stronger password.
```

Example 3:

```
Please enter your email: example@email.com
Please enter a password: PLEXMAILPASS
Password is too similar to email. Please choose a different password.
```

Nädal 3. Praktikumi ülesanne 1 (Exercise 1 - Combinations)

Write a program that asks the user for two inputs. Output unique two-letter combinations where the first letter is from the first input and the second letter is from the second input. All of the unique letters should be used.

Choose the data structure that is best suited for keeping unique letters and combinations from the inputs.

Example:

```
Enter something: Hello
Enter something: Goodbye
```

Output:

```
{'le', 'Ho', 'He', 'lb', 'ly', 'Hd', 'lG', 'eG', 'eo', 'oG', 'lo', 'HG',
'oo', 'od', 'oy', 'oe', 'ld', 'Hy', 'ob', 'ey', 'Hb', 'ee', 'ed', 'eb'}
```

Nädal 3. Praktikumi ülesanne 2 (Exercise 2 - Shopping bills - unique items)

The file "bills.txt" contains bills for 1 week of shopping, where each line represents one shopping bill.

Example bills.txt file:

```
bread,butter,milk,chocolate
onions,milk,sugar
garlic,pepper,salt,milk,bread,candy
```

Write a program that reads in the file and outputs all the items that were bought during that week. Output each item only once.

Output:

```
{'onions', 'milk', 'sugar', 'chocolate', 'candy', 'salt', 'garlic',
'pepper', 'bread', 'butter'}
```

Nädal 3. Praktikumi ülesanne 3 (Exercise 3 - Shopping bills - common items)

Modify the previous program (Session 3 Exercise 2) so that it outputs all the items that were bought each time.

Output each item only once.

Example bills.txt file:

```
bread,butter,milk,chocolate
onions,milk,sugar
garlic,pepper,salt,milk,bread,candy
```

Output:

```
{'milk'}
```

Nädal 3. Praktikumi ülesanne 4 (Exercise 4 - BMI)

Information about the people in your class, such as their gender, age, weight, height, and identification number, is stored in a list of tuples.

Example:

```
classmates= [ ( "male", 21, 82,  185, 50002250426),
              ( "male", 22, 74, 178, 50003210406),
              ( "female", 22, 68, 172, 60008220306),
              ( "female", 23, 64, 171, 600122501124)]
```

To calculate the body mass index (BMI), there are two formulas:

- **Old formula: BMI = weight(kg)/height(m)^2**
- **New formula: BMI = 1.3*weight(kg)/height(m)^2.5**

Write a function that calculates the difference in average body mass index (BMI) in class between the two formulas and returns the difference, the average BMI according to the new formula, and whether it is a good result or not. According to the Centers for Disease Control and Prevention, a healthy BMI range is between 18.5 and 24.9.

Hint: Use the built-in *abs()* function to find the absolute value of difference.

Example output:

```
>>>calculate_BMI(classmates)
BMI Difference: 0.50
Average BMI according to new formula: 22.55
The average BMI in class is in a healthy range!
```

Nädal 3. Praktikumi ülesanne 5 (Exercise 5)

When converting from one currency to another, you might need to use a third currency in the middle. Write a function that takes a list of tuples and an integer as inputs and returns the best conversion through a third currency from 'USD' to 'EUR', and the amount of euros you will have.

The rates are in tuples in the format:

- (US Dollar currency code, USD to a third currency rate, third currency code, third currency to EUR rate, EURO currency code)

For example:

```
rates = [('USD',0.7969299999999999, 'GBP', 1.140575, 'EUR'),
         ('USD',1.354685, 'CAD', 0.6694908683, 'EUR'),
         ('USD',1.50688, 'AUD', 0.60349, 'EUR'),
         ('USD',136.322, 'JPY', 0.0066300000000000005, 'EUR'),
         ('USD',0.89222, 'CHF', 1.016722189, 'EUR'),
         ('USD',7.84995, 'HKD', 0.11555575280000001, 'EUR'),
         ('USD',1.616515, 'NZD', 0.5626749999999999, 'EUR'),
         ('USD',1.331745, 'SGD', 0.6811538115, 'EUR')]
```

Iterate through the tuples and find the currency with the best rates so that, by the end of the conversion, you will have the most euros.

The formula to calculate the conversion is:

- Amount * USD to Third currency rate * Third currency to EUR rate

Output the currency that is the most profitable and how many euros you will have.

Example:

```
>>>best_conversion(rates, 100)
The best currency to convert USD to EUR is NZD.
You'll receive 90.96 EUR.
```

Nädal 3. Praktikumi ülesanne 6 (Exercise 6 - Data types)

Write a function that takes a tuple containing elements of different data types and returns a tuple containing all the data types present in the input tuple.

Hints:

- Use the *type(element)* function to check the data type of each element.
- Use *set()* to find each data type only once.

Example tuple:

```
elements = ("apple", 10, True, 3.14, "orange", 20, False, 2.71, "banana",
30, True, 1.618, [1, 2, 3], {"name": "John", "age": 30})
```

Output:

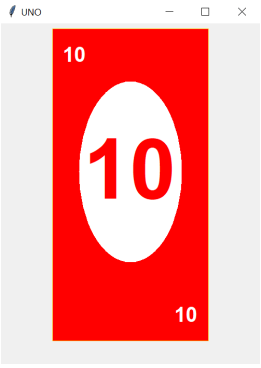
```
(<class 'float'>, <class 'bool'>, <class 'int'>, <class 'list'>, <class
'str'>, <class 'dict'>)
```

Nädal 4. Kodutöö 1 (Homework 1 - UNO)

"UNO" is a popular card game that many of us have played in our youth. Choose a UNO card and write a program that uses Tkinter and its canvas widget to draw the chosen card in the window with a white background. The title of the window should be called "UNO".

UNO cards: https://en.wikipedia.org/wiki/Uno_(card_game)

Example UNO drawing:



Nädal 4. Kodutöö 2 (Homework 2 - License plate)

In Estonia, a car's license plate usually consists of three numbers and three capital letters, for example, 123ABC. The file "plates.txt" contains information about cars such as the year, brand, color, and license plate numbers in a specific format, where a specific letter indicates the type of information:

- y - year
- b - brand
- c - color
- l - license plate number

All information is in capital letters. Your task is to output the cars that have Estonian license plate numbers.

Example plates.txt file:

```
y2009bVOLVOcGRAYl123ABC
y2010bSAABcGRAYl133GBC
y2002bBMWcYELLOWl23ABC
y2015bAUDIcWHITElBG-432
```

Output:

```
('2009', 'VOLVO', 'GREY', '123ABC')
('2010', 'SAAB', 'GREY', '133GBC')
```

Nädal 4. Praktikumi ülesanne 1 (Exercise 1 - Major events)

File "major_events.txt" contains dates of major events in recent history. The dates are from the website https://www.timetoast.com/timelines/major-events-in-the-last-100-years--2

Your task is to write a program that reads the dates from the file and draws a bar chart based on the number of times an event happened in each month (use Tkinter and the canvas).

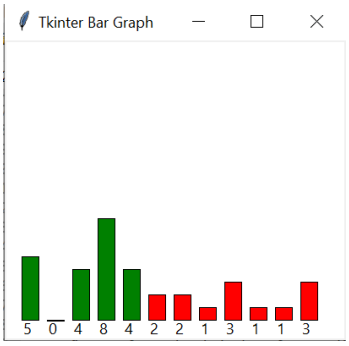
The columns with values equal to or greater than 4 should be filled with green. Otherwise, the columns should be in red. The values have to be added under the columns, and the bars have to go according to months: January, February, March, etc.

Hint: read the file into the dictionary first

The major_events.txt file:

15.04.1920	21.09.1920	27.07.1921	05.12.1924
20.05.1927	22.01.1930	04.09.1930	01.03.1933
16.06.1933	14.04.1935	30.01.1940	07.12.1941
08.06.1943	16.04.1945	08.05.1945	01.11.1955
12.04.1961	11.08.1962	02.07.1964	15.01.1967
24.05.1977	28.03.1979	19.04.1980	13.05.1981
08.03.1983	20.01.1993	19.04.1995	20.04.1999
31.12.1999	11.09.2001	02.10.2002	20.03.2003
16.04.2007	15.01.2009		

Output:



Nädal 4. Praktikumi ülesanne 2 (Exercise 2 - Phone numbers)

Write a function that uses regular expressions to make sure a given phone number is written correctly.

Phone number structure:

- Phone number length <= 4: XXXX
- Phone number length = 5: XX XXX
- Phone number length = 6: XXX XXX
- Phone number length = 7: XXX XXXX
- Phone number length = 8: XXXX XXXX
- To separate number groups, '-' can be used instead of space
- Phone numbers may include a country code with a '+' at the beginning.

Multiple regular expressions may be needed to check the phone number.

Example 1:

```
>>>phone_number("+123 5859 9399")
Correct!
```

Example 2:

```
>>>phone_number("585-9399")
Correct!
```

Example 3:

```
>>>phone_number("8599399")
Incorrect!
```

Nädal 4. Praktikumi ülesanne 4 (Exercise 4 - Rainbow)

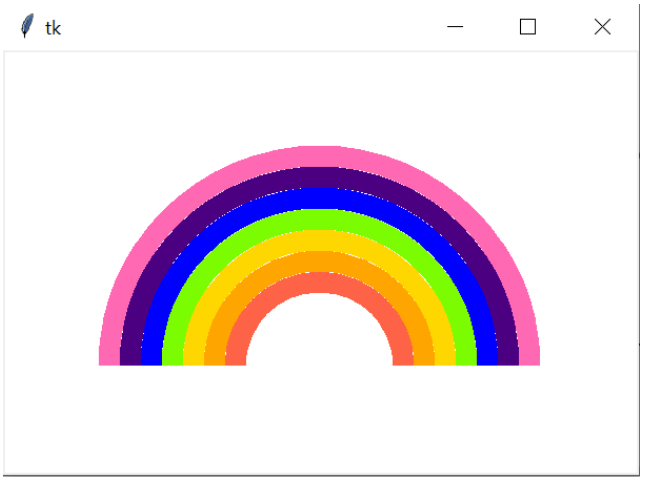
You can find a list of color names that the Tkinter module recognizes from http://wiki.tcl.tk/16166.

Your task is to choose 7 colors and draw a rainbow using those colors with Tkinter.

Example:

```
colors = ['hot pink', 'indigo', 'blue', 'lawn green', 'gold', 'orange',
'tomato']
```

Output:



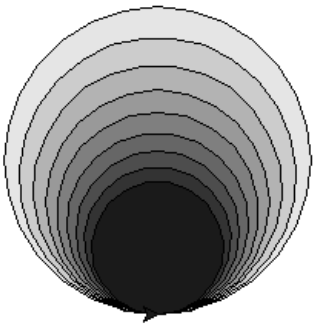
Nädal 5. Kodutöö 1 (Homework 1 - Tunnel)

Write a function that takes the depth, radius and color as parameters and draws a tunnel using turtle. With each iteration, the tunnel has to get darker in color. Use the color ‘gray’ variations (*gray0, gray10, gray20 ...*) from this website: https://cs111.wellesley.edu/labs/lab02/colors.

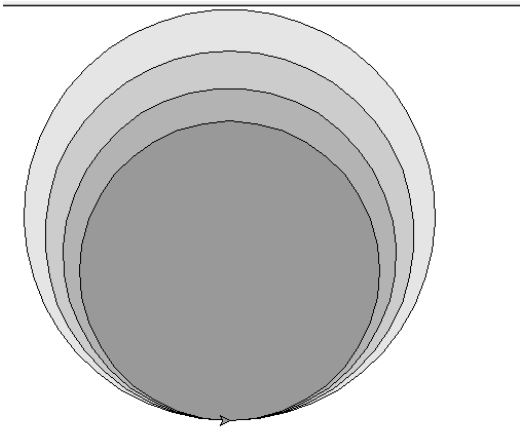
Hint:

- Use ***t.fillcolor("black")*** to adjust color.
- Use ***t.begin_fill()*** and ***t.end_fill()*** to fill an object with color.

Example tunnel with depth 10 and radius 100:



Example tunnel with depth 5 and radius 200:



Nädal 5. Kodutöö 2 (Homework 2 - Division)

Write a recursive function that takes two numbers (dividend and divisor) as arguments and prints the result of dividing the first argument by the second argument until the dividend is less than 1.

Example:

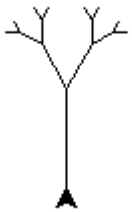
```
>>>div(29,3)
9.666666666666666
3.222222222222222
1.074074074074074
0.35802469135802467
```

Nädal 5. Praktikumi ülesanne 1 (Exercise 1 - Fractal tree)

A Y-shaped fractal tree is one where each branch splits into two smaller branches, and each of those branches also splits into two more. Write a recursive function that takes the tree height and the number of levels for sub-branches and uses turtle to draw the tree.

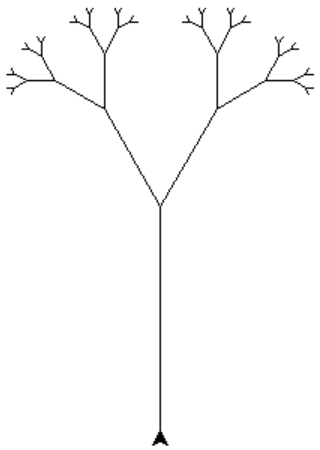
Example 1:

```
>>>fractal_tree(50, 4)
```



Example 2:

```
>>>fractal_tree(150, 6)
```



Nädal 5. Praktikumi ülesanne 2 (Exercise 2 - Multiplication)

Write a recursive function *multiply()* that takes two numbers as its arguments and returns their multiplication.

Note: The use of * is not allowed.

Example 1:

```
>>>multiply(4,5)
20
```

Example 2:

```
>>>multiply(7,6)
42
```

Nädal 5. Praktikumi ülesanne 3 (Exercise 3 - Reverse order of odd numbers)

Write a recursive function called *reverse()* that takes a list of numbers as input. The function should iterate through the list in reverse order and return a new list containing only the positive odd numbers from the input list.

Example 1:

```
>>>reverse([2, -5, 7, 0, -7, 1])
[1,7]
```

Example 2:

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

Nädal 5. Praktikumi ülesanne 4 (Exercise 4 - Smallest number in the list)

Write a recursive function *findMin()* that takes a recursive data structure (a nested list) and returns the smallest number in all lists.

Hint:

- Use the built-in *min()* function to compare numbers
- Use the built-in *type()* to check the object type

Example 1:

```
>>>findMin([3, [2, 3], [[[[4, 5], 1]]], 7, 8])
1
```

Example 2:

```
>>>findMin([11, [1, 3], [[[[-1, 2], 10]]], 3, 2])
-1
```

III. Koostatud murelahendajad

Nädal 1. Kodutöö 1. : https://progtugi.cs.ut.ee/#/ts/64215ad2953066fc5ca1cfac/

Nädal 2. Kodutöö 1. : https://progtugi.cs.ut.ee/#/ts/642525e3953066fc5ca1d79c/

Nädal 2. Kodutöö 2. : https://progtugi.cs.ut.ee/#/ts/6423f2eb953066fc5ca1d406/

Nädal 3. Kodutöö 1. : https://progtugi.cs.ut.ee/#/ts/642171c9953066fc5ca1d00a/

Nädal 3. Kodutöö 2. : https://progtugi.cs.ut.ee/#/ts/64251330953066fc5ca1d689/

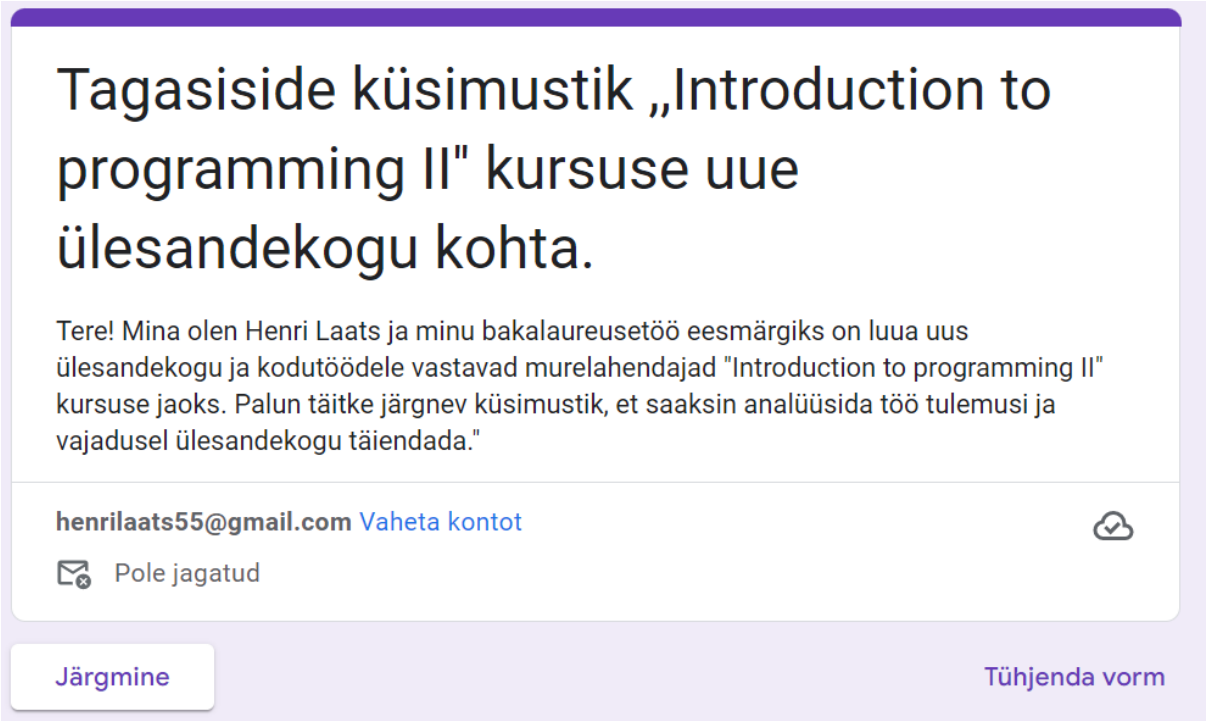
Nädal 4. Kodutöö 1. : https://progtugi.cs.ut.ee/#/ts/642183fb953066fc5ca1d02b/

Nädal 4. Kodutöö 2. : https://progtugi.cs.ut.ee/#/ts/64218d9f953066fc5ca1d055/

Nädal 5. Kodutöö 1. : https://progtugi.cs.ut.ee/#/ts/642198d4953066fc5ca1d06c/

Nädal 5. Kodutöö 2. : https://progtugi.cs.ut.ee/#/ts/6421a0c5953066fc5ca1d0a0/

IV. Ülesandekogu tagasiside küsimustik



Tagasiside küsimustik „Introduction to programming II" kursuse uue ülesandekogu kohta.

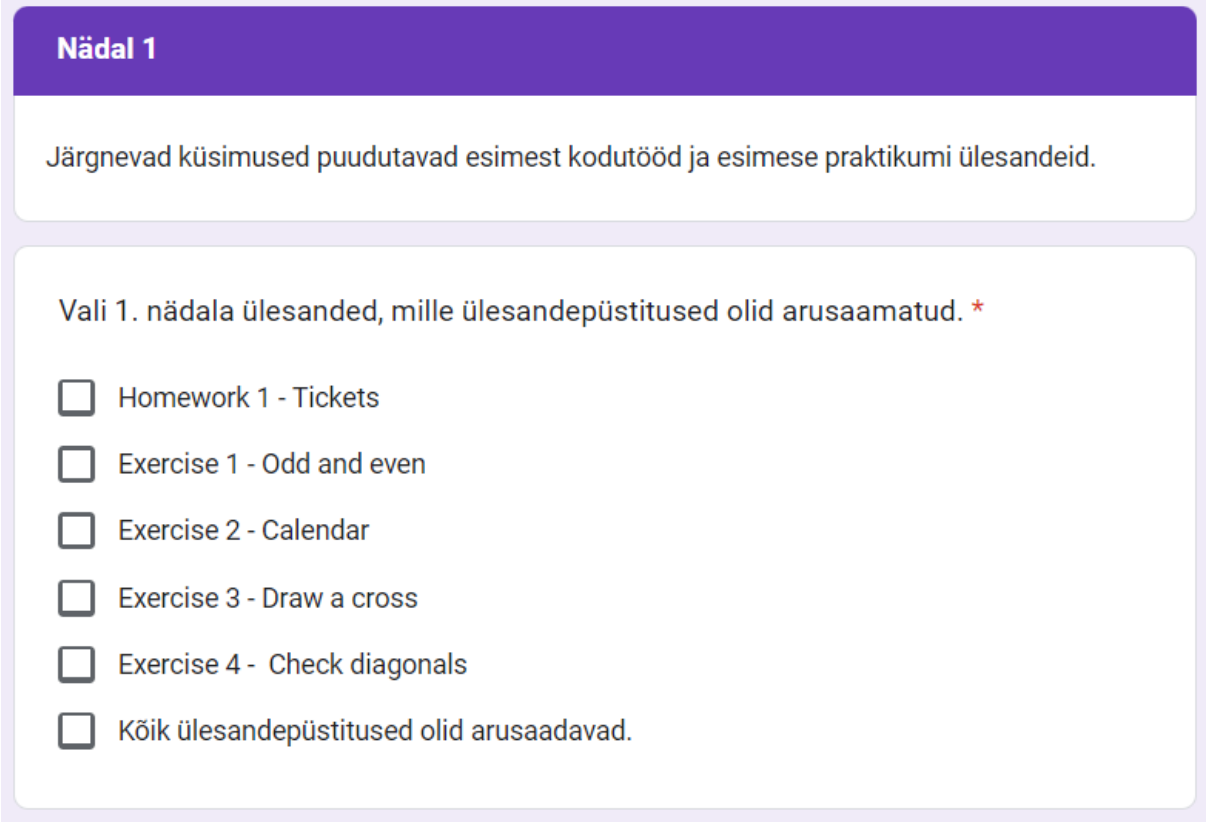
Tere! Mina olen Henri Laats ja minu bakalaureusetöö eesmärgiks on luua uus ülesandekogu ja kodutöödele vastavad murelahendajad "Introduction to programming II" kursuse jaoks. Palun täitke järgnev küsimustik, et saaksin analüüsida töö tulemusi ja vajadusel ülesandekogu täiendada."

henrilaats55@gmail.com Vaheta kontot

Pole jagatud

Järgmine

Tühjenda vorm



Nädal 1

Järgnevad küsimused puudutavad esimest kodutööd ja esimese praktikumi ülesandeid.

Vali 1. nädala ülesanded, mille ülesandepüstitused olid arusaamatud. *

- [ ] Homework 1 - Tickets
- [ ] Exercise 1 - Odd and even
- [ ] Exercise 2 - Calendar
- [ ] Exercise 3 - Draw a cross
- [ ] Exercise 4 - Check diagonals
- [ ] Kõik ülesandepüstitused olid arusaadavad.

Vali 1. nädala ülesanded, mis keskenduvad piisavalt nädala põhiteemale (Topelt tsükkel). *

- [ ] Homework 1 - Tickets
- [ ] Exercise 1 - Odd and even
- [ ] Exercise 2 - Calendar
- [ ] Exercise 3 - Draw a cross
- [ ] Exercise 4 - Check diagonals
- [ ] Ükski ülesanne ei olnud piisavalt keskendunud selle nädala põhiteemale.

Vali 1. nädala ülesanded, mis olid liiga keerulised. *

- [ ] Homework 1 - Tickets
- [ ] Exercise 1 - Odd and even
- [ ] Exercise 2 - Calendar
- [ ] Exercise 3 - Draw a cross
- [ ] Exercise 4 - Check diagonals
- [ ] Ükski ülesanne polnud liiga keeruline.

Vali 1. nädala ülesanded, mis võiksid olla keerulisemad. *

- [ ] Homework 1 - Tickets
- [ ] Exercise 1 - Odd and even
- [ ] Exercise 2 - Calendar
- [ ] Exercise 3 - Draw a cross
- [ ] Exercise 4 - Check diagonals
- [ ] Kõik ülesanded olid piisavalt keerulised.

Vali 1. nädala ülesanded, mis võiksid tudengeid rohkem kõnetada. *

- [ ] Homework 1 - Tickets
- [ ] Exercise 1 - Odd and even
- [ ] Exercise 2 - Calendar
- [ ] Exercise 3 - Draw a cross
- [ ] Exercise 4 - Check diagonals
- [ ] Kõik ülesanded olid piisavalt asjakohased.

Kommentaarid 1. nädala ülesandekogu kohta.

Teie vastus

Tagasi Järgmine Tühjenda vorm

Nädal 2

Järgnevad küsimused puudutavad teist kodutööd ja teise praktikumi ülesandeid.

Vali 2. nädala ülesanded, mille ülesandepüstitused olid arusaamatud. *

- [ ] Homework 1 - Netflix
- [ ] Homework 2 - Electricity bill
- [ ] Exercise 1 - Input to Dictionary of Letters
- [ ] Exercise 2 - Workout
- [ ] Exercise 3 - Fridge & recipes
- [ ] Kõik ülesandepüstitused olid arusaadavad.

Vali 2. nädala ülesanded, mis keskenduvad piisavalt nädala põhiteemale (Sõnastikud). *

- [ ] Homework 1 - Netflix
- [ ] Homework 2 - Electricity bill
- [ ] Exercise 1 - Input to Dictionary of Letters
- [ ] Exercise 2 - Workout
- [ ] Exercise 3 - Fridge & recipes
- [ ] Ükski ülesanne ei olnud piisavalt keskendunud selle nädala põhiteemale.

Vali 2. nädala ülesanded, mis olid liiga keerulised. *

- [ ] Homework 1 - Netflix
- [ ] Homework 2 - Electricity bill
- [ ] Exercise 1 - Input to Dictionary of Letters
- [ ] Exercise 2 - Workout
- [ ] Exercise 3 - Fridge & recipes
- [ ] Ükski ülesanne polnud liiga keeruline.

Vali 2. nädala ülesanded, mis võiksid olla keerulisemad. *

- [ ] Homework 1 - Netflix
- [ ] Homework 2 - Electricity bill
- [ ] Exercise 1 - Input to Dictionary of Letters
- [ ] Exercise 2 - Workout
- [ ] Exercise 3 - Fridge & recipes
- [ ] Kõik ülesanded olid piisavalt keerulised.

Vali 2. nädala ülesanded, mis võiksid tudengeid rohkem kõnetada. *

- [ ] Homework 1 - Netflix
- [ ] Homework 2 - Electricity bill
- [ ] Exercise 1 - Input to Dictionary of Letters
- [ ] Exercise 2 - Workout
- [ ] Exercise 3 - Fridge & recipes
- [ ] Kõik ülesanded olid piisavalt asjakohased.

Kommentaarid 2. nädala ülesandekogu kohta.

Teie vastus

Tagasi Järgmine Tühjenda vorm

Nädal 3

Järgnevad küsimused puudutavad kolmandat kodutööd ja kolmanda praktikumi ülesandeid.

Vali 3. nädala ülesanded, mille ülesandepüstitused olid arusaamatud. *

- [ ] Homework 1 - TOP 3 letters
- [ ] Homework 2 - Passwords
- [ ] Exercise 1 - Combinations
- [ ] Exercise 2 - Shopping bills - unique items
- [ ] Exercise 3 - Shopping bills - common items
- [ ] Exercise 4 - Classmates
- [ ] Exercise 5 - Team captains
- [ ] Exercise 6 - Days left
- [ ] Kõik ülesandepüstitused olid arusaadavad.

Vali 3. nädala ülesanded, mis keskenduvad piisavalt nädala põhiteemale (Ennikud, hulgad). *

- [ ] Homework 1 - TOP 3 letters
- [ ] Homework 2 - Passwords
- [ ] Exercise 1 - Combinations
- [ ] Exercise 2 - Shopping bills - unique items
- [ ] Exercise 3 - Shopping bills - common items
- [ ] Exercise 4 - Classmates
- [ ] Exercise 5 - Team captains
- [ ] Exercise 6 - Days left
- [ ] Ükski ülesanne ei olnud piisavalt keskendunud selle nädala põhiteemale.

Vali 3. nädala ülesanded, mis olid liiga keerulised. *

- [ ] Homework 1 - TOP 3 letters
- [ ] Homework 2 - Passwords
- [ ] Exercise 1 - Combinations
- [ ] Exercise 2 - Shopping bills - unique items
- [ ] Exercise 3 - Shopping bills - common items
- [ ] Exercise 4 - Classmates
- [ ] Exercise 5 - Team captains
- [ ] Exercise 6 - Days left
- [ ] Ükski ülesanne polnud liiga keeruline.

Vali 3. nädala ülesanded, mis võiksid olla keerulisemad. *

- [ ] Homework 1 - TOP 3 letters
- [ ] Homework 2 - Passwords
- [ ] Exercise 1 - Combinations
- [ ] Exercise 2 - Shopping bills - unique items
- [ ] Exercise 3 - Shopping bills - common items
- [ ] Exercise 4 - Classmates
- [ ] Exercise 5 - Team captains
- [ ] Exercise 6 - Days left
- [ ] Kõik ülesanded olid piisavalt keerulised.

Vali 3. nädala ülesanded, mis võiksid tudengeid rohkem kõnetada. *

- [ ] Homework 1 - TOP 3 letters
- [ ] Homework 2 - Passwords
- [ ] Exercise 1 - Combinations
- [ ] Exercise 2 - Shopping bills - unique items
- [ ] Exercise 3 - Shopping bills - common items
- [ ] Exercise 4 - Classmates
- [ ] Exercise 5 - Team captains
- [ ] Exercise 6 - Days left
- [ ] Kõik ülesanded olid piisavalt asjakohased.

Kommentaarid 3. nädala ülesandekogu kohta.

Teie vastus

Tagasi Järgmine Tühjenda vorm

Nädal 4

Järgnevad küsimused puudutavad neljandat kodutööd ja neljanda praktikumi ülesandeid.

Vali 4. nädala ülesanded, mille ülesandepüstitused olid arusaamatud. *

- [ ] Homework 1 - UNO
- [ ] Homework 2 - License plate
- [ ] Exercise 1 - Major events
- [ ] Exercise 2 - Phone numbers
- [ ] Exercise 4 - Rainbow
- [ ] Kõik ülesandepüstitused olid arusaadavad.

Vali 4. nädala ülesanded, mis keskenduvad piisavalt nädala põhiteemale (Graafika, Regulaaravaldised). *

- [ ] Homework 1 - UNO
- [ ] Homework 2 - License plate
- [ ] Exercise 1 - Major events
- [ ] Exercise 2 - Phone numbers
- [ ] Exercise 4 - Rainbow
- [ ] Ükski ülesanne ei olnud piisavalt keskendunud selle nädala põhiteemale.

Vali 4. nädala ülesanded, mis olid liiga keerulised. *

- [ ] Homework 1 - UNO
- [ ] Homework 2 - License plate
- [ ] Exercise 1 - Major events
- [ ] Exercise 2 - Phone numbers
- [ ] Exercise 4 - Rainbow
- [ ] Ükski ülesanne polnud liiga keeruline.

Vali 4. nädala ülesanded, mis võiksid olla keerulisemad. *

- [ ] Homework 1 - UNO
- [ ] Homework 2 - License plate
- [ ] Exercise 1 - Major events
- [ ] Exercise 2 - Phone numbers
- [ ] Exercise 4 - Rainbow
- [ ] Kõik ülesanded olid piisavalt keerulised.

Vali 4. nädala ülesanded, mis võiksid tudengeid rohkem kõnetada. *

- [ ] Homework 1 - UNO
- [ ] Homework 2 - License plate
- [ ] Exercise 1 - Major events
- [ ] Exercise 2 - Phone numbers
- [ ] Exercise 4 - Rainbow
- [ ] Kõik ülesanded olid piisavalt asjakohased.

Kommentaarid 4. nädala ülesandekogu kohta.

Teie vastus

Tagasi Järgmine Tühjenda vorm

Nädal 5

Järgnevad küsimused puudutavad viiendat kodutööd ja viienda praktikumi ülesandeid.

Vali 5. nädala ülesanded, mille ülesandepüstitused olid arusaamatud. *

- [ ] Homework 1 - Tunnel
- [ ] Homework 2 - Division
- [ ] Exercise 1 - Fractal tree
- [ ] Exercise 2 - Sierpinski triangle
- [ ] Exercise 3 - Multiplication
- [ ] Exercise 4 - Smallest number in the list
- [ ] Kõik ülesandepüstitused olid arusaadavad.

Vali 5. nädala ülesanded, mis keskenduvad piisavalt nädala põhiteemale (Rekursioon). *

- [ ] Homework 1 - Tunnel
- [ ] Homework 2 - Division
- [ ] Exercise 1 - Fractal tree
- [ ] Exercise 2 - Sierpinski triangle
- [ ] Exercise 3 - Multiplication
- [ ] Exercise 4 - Smallest number in the list
- [ ] Ükski ülesanne ei olnud piisavalt keskendunud selle nädala põhiteemale.

Vali 5. nädala ülesanded, mis olid liiga keerulised. *

- [ ] Homework 1 - Tunnel
- [ ] Homework 2 - Division
- [ ] Exercise 1 - Fractal tree
- [ ] Exercise 2 - Sierpinski triangle
- [ ] Exercise 3 - Multiplication
- [ ] Exercise 4 - Smallest number in the list
- [ ] Ükski ülesanne polnud liiga keeruline.

Vali 5. nädala ülesanded, mis võiksid olla keerulisemad. *

- [ ] Homework 1 - Tunnel
- [ ] Homework 2 - Division
- [ ] Exercise 1 - Fractal tree
- [ ] Exercise 2 - Sierpinski triangle
- [ ] Exercise 3 - Multiplication
- [ ] Exercise 4 - Smallest number in the list
- [ ] Kõik ülesanded olid piisavalt keerulised.

Vali 5. nädala ülesanded, mis võiksid tudengeid rohkem kõnetada. *

- [ ] Homework 1 - Tunnel
- [ ] Homework 2 - Division
- [ ] Exercise 1 - Fractal tree
- [ ] Exercise 2 - Sierpinski triangle
- [ ] Exercise 3 - Multiplication
- [ ] Exercise 4 - Smallest number in the list
- [ ] Kõik ülesanded olid piisavalt asjakohased.

Kommentaarid 5. nädala ülesandekogu kohta.

Teie vastus

V. Murelahendajate tagasiside küsimustik

Tagasiside küsimustik kursusele „Introduction to programming II" loodud murelahendajate kohta.

Tere!
Mina olen Henri Laats ja minu bakalaureusetöö eesmärgiks on luua uus ülesandekogu ja kodutöödele vastavad murelahendajad "Introduction to programming II" kursuse jaoks. Palun täitke järgnev küsimustik, et saaksin analüüsida loodud murelahendajaid ning neid vajaduse korral täiendada.

henrilaats55@gmail.com Vaheta kontot

Pole jagatud

* Viitab kohustuslikule küsimusele

Märgi murelahendajad, mis on arvestanud enamike võimalike keeruliste kohtadega. *

- [ ] Week 1 - Homework 1 - Average grade
- [ ] Week 2 - Homework 1 - Travelling
- [ ] Week 2 - Homework 2 - Electricity bill
- [ ] Week 3 - Homework 1 - Top 3 letters
- [ ] Week 3 - Homework 2 - Password strength checker
- [ ] Week 4 - Homework 1 - UNO
- [ ] Week 4 - Homework 2 - License plate
- [ ] Week 5 - Homework 1 - Tunnel
- [ ] Week 5 - Homework 2 - Division
- [ ] Mitte ükski

Märgi murelahendajad, mis aitavad keeruliste kohtade lahendamisega. *

- [ ] Week 1 - Homework 1 - Average grade
- [ ] Week 2 - Homework 1 - Travelling
- [ ] Week 2 - Homework 2 - Electricity bill
- [ ] Week 3 - Homework 1 - Top 3 letters
- [ ] Week 3 - Homework 2 - Password strength checker
- [ ] Week 4 - Homework 1 - UNO
- [ ] Week 4 - Homework 2 - License plate
- [ ] Week 5 - Homework 1 - Tunnel
- [ ] Week 5 - Homework 2 - Division
- [ ] Mitte ükski

Märgi murelahendajad, mis annavad liiga palju informatsiooni ning võivad anda vastuse ette." *

- [ ] Week 1 - Homework 1 - Average grade
- [ ] Week 2 - Homework 1 - Travelling
- [ ] Week 2 - Homework 2 - Electricity bill
- [ ] Week 3 - Homework 1 - Top 3 letters
- [ ] Week 3 - Homework 2 - Password strength checker
- [ ] Week 4 - Homework 1 - UNO
- [ ] Week 4 - Homework 2 - License plate
- [ ] Week 5 - Homework 1 - Tunnel
- [ ] Week 5 - Homework 2 - Division
- [ ] Mitte ükski

Märgi murelahendajad, mille sõnastus vajaks täiendamist. *

- [ ] Week 1 - Homework 1 - Average grade
- [ ] Week 2 - Homework 1 - Travelling
- [ ] Week 2 - Homework 2 - Electricity bill
- [ ] Week 3 - Homework 1 - Top 3 letters
- [ ] Week 3 - Homework 2 - Password strength checker
- [ ] Week 4 - Homework 1 - UNO
- [ ] Week 4 - Homework 2 - License plate
- [ ] Week 5 - Homework 1 - Tunnel
- [ ] Week 5 - Homework 2 - Division
- [ ] Mitte ükski

Kommentaarid
Järgnevalt on võimalik iga murelahendaja kohta jätta täiendavaid märkusi.

Kommentaarid ***Week 1 - Homework 1 - Average grade*** murelahendaja kohta

Teie vastus

Kommentaarid ***Week 2 - Homework 1 - Travelling*** murelahendaja kohta

Teie vastus

Kommentaarid ***Week 2 - Homework 2 - Electricity bill*** murelahendaja kohta

Teie vastus

Kommentaarid ***Week 3 - Homework 1 - Top 3 letters*** murelahendaja kohta

Teie vastus

Kommentaarid ***Week 3 - Homework 2 - Password strength checker*** murelahendaja kohta

Teie vastus

Kommentaarid ***Week 4 - Homework 1 - UNO*** murelahendaja kohta

Teie vastus

Kommentaarid ***Week 4 - Homework 2 - License plate*** murelahendaja kohta

Teie vastus

Kommentaarid ***Week 5 - Homework 1 - Tunnel*** murelahendaja kohta

Teie vastus

Kommentaarid ***Week 5 - Homework 2 - Division*** murelahendaja kohta

Teie vastus

Saada ära

Tühjenda vorm

VI. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Henri Laats,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose Uue ülesandekogu ja kodutöödele vastavate murelahendajate koostamine kursusele ,,Introduction to programming II'', mille juhendaja on Reelika Suviste, 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 4.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, alates **09.05.2023** 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.

Henri Laats
22.04.2023