TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Heli-Katri Marttila

Programmide sarnasuse määratlemine ühe programmeerimisülesande näitel

Bakalaureusetöö (9 EAP)

Juhendajad:
Eno Tõnisson
Heidi Meier

Tartu 2020

Programmide sarnasuse määratlemine ühe programmeerimisülesande näitel

Lühikokkuvõte:

Programmeerimise õppimine ja õpetamine muutub aina populaarsemaks. Kuna seda tehakse erinevates õppevormides ja erinevate juhendajatega, võivad koostööle ja materjalide kasutamisele kehtestatud nõuded suuresti erineda. Seetõttu võivad keerukamate situatsioonide puhul erineda õpetajate ja õppijate arusaamad lubatud sarnasusest ja plagiaadist. Kuigi programmide sarnasuse tuvastamiseks saab kasutada automaatseid vahendeid, peab lõpliku otsuse tegema siiski inimene. Töös uuriti algajatele mõeldud programmeerimiskursuse ühe ülesande 24 lahenduse omavahelist sarnasust. Seda tehti kolmes faasis, esialgu vaadati programmid käsitsi läbi, seejärel kasutati Moodle'i automaatset sarnasuse analüüsi, mõlema tulemusena saadi sarnaste lahenduste klastrid. Kolmandaks moodustati eelneva kahe tulemuste põhjal kuus olemuselt erinevat lahenduste paari, mida paluti hinnata sama kursuse mentoritel. Selle tulemusena selgus, milliseid aspekte peetakse sarnasuse hindamisel peamiselt silmas ning milliseid tehnikaid võidakse jälgede segamiseks kasutada. Esile toodi muutujate nimed, lahenduste pikkused, programmide struktuur ja sisu. Mentorite üldised hinnangud plagiaadi osas peamiselt kattusid, kuid esines ka vastandlikke arvamusi.

Võtmesõnad:

Programmeerimine, programmeerimise õpetamine, lahenduste sarnasus.

CERCS: P175 Informaatika, süsteemiteooria

Determining Code Similarity Using One Programming Task

Abstract:

Teaching and studying programming is becoming increasingly popular. Since it can be done in a variety of forms and facilitated by different people, the requirements set on co-working and the use of materials may also vary. This is why students' and teachers' perception of allowed similarity and plagiarism may vary when considering complex situations. Although there are automated applications for similarity analysis, the ultimate decision still lies on a human. In this thesis, 24 solutions to one programming task were analysed for their similarity. The solutions were submitted by the participants of a programming course meant for beginners. The analysis was done in three steps – firstly the codes were analysed manually, then Moodle's automated similarity analysis was used. Both methods resulted in clusters of similar solutions. Based on these results, as a third step, six pairs of solutions were constructed. These pairs were evaluated by the mentors of the programming course. As a result, the main aspects that are checked when assessing similarity, namely the variables, the length of the code, the structure and the content of the solutions were explained. The main techniques that can be used to hide traces of plagiarism were also included. The mentors' views generally overlapped, but there were also contrary opinions.

Keywords:

Programming, teaching programming, code similarity

CERCS: P175 Informatics, systems theory

Sisukord

1. Sissejuhatus

Digitaliseeruvas maailmas on üha suurem roll tehnoloogial. Seetõttu on loomulik, et ka programmeerimine ning selle õppimine ja õpetamine muutub jätkuvalt populaarsemaks. Õppida on võimalik erinevates vormides nii koolides kui ka vaid veebi vahendusel. Õppuriteks võivad olla nii põhikooli õpilased kui ka pensioniealised. Ka õppijatele loodud tingimused ja nõuded võivad isegi ühes õppeasutuses erinevate ainete puhul suuresti erineda.

Praktilise olemuse tõttu on programmeerimise õppimisel oluline osa ülesannete lahendamisel. Seetõttu on üks oluline aspekt programmeerimise õpetamise juures õppijate esitatud programmide ja nende nõuetele vastavuse hindamine, seejuures ka programmide sarnasuse hindamine. Viimane ülesanne osutub küllaltki keerukaks, kuna lubatud sarnasuse määratlemine on võrdlemisi keeruline. Ka pahatahtliku soovi puudumisel võib õppurites tekitada segadust näiteks see, et programmeerimise puhul üldiselt propageeritakse erinevate meediumite vahendusel informatsiooni otsimist ja olemasolevate koodiridade kasutamist, mistõttu lubatud käitumise piirid võivad muutuda hägusamaks (Joy *et al.*, 2010).

Bakalaureusetöö eesmärk on tutvustada programmide sarnasuse analüüsimise probleemi ning näidata, kuidas sarnasust hinnatakse ning millistel juhtudel ja millistest aspektidest tulenevalt plagiaati kahtlustatakse. Töö praktilises osas analüüsitakse Tartu Ülikooli programmeerimise e-kursuse "Tehnoloogia tarbijast loojaks" osalejate ühe ülesande lahendusi. Tegemist on 10 nädalat kestva 16–26-aastastele algajatele mõeldud kursusega.

Töö on jaotatud kolmeks peatükiks. Esimeses peatükis tutvustatakse programmide sarnasuse analüüsi keerukuse põhjuseid ja kirjeldatakse erinevusi õppijate ja õpetajate perspektiivides. Sellest tulenevalt uuritakse selles töös, kuivõrd on erisusi mentorite hinnangutes plagiaadi osas ja selgitatakse lahenduste sarnasuse võimalikke põhjuseid.

Teises peatükis selgitatakse autoripoolset ühe ülesande lahenduste uurimise käiku ja tulemusi. Seda tehakse kahes faasis, esmalt tutvustatakse tööde käsitsi läbivaatamise metoodikat ja tulemusi, seejärel Moodle'i automaatkontrolli vahendi kasutamisel saadud tulemusi. Lõpuks võrreldakse kummagi meetodi kaudu saadud sarnaste lahenduste klastreid.

Kolmandas peatükis analüüsitakse kursuse "Tehnoloogia tarbijast loojaks" mentoritelt küsitluse vahendusel saadud hinnanguid kuuele lahenduste paarile. Küsitluses paluti hinnata, kuivõrd on tegemist plagiaadiga, tuua argumente plagiaadi poolt ja vastu ning selgitada, mida

võib olla tehtud jälgede peitmiseks. Vastuseid vaadeldakse kõigepealt iga paari kohta eraldi ning lõpuks näidatakse kõikide paaride peale saadud tulemuste põhjal, millised on peamised aspektid, mida hindamisel jälgitakse, ning kuidas need plagiaadi poolt või vastu võivad rääkida.

Terminitest tuleb kommenteerida kahte. Kogu töö ulatuses kasutatakse terminit “akadeemiline petturlus” (ingl *academic offense)* isegi kui õigusaktide järgi ei pruugi täpselt sellega tegemist olla. Teine võimalik variant oleks “akadeemilise tava rikkumine”. Töös hinnatakse küllaltki lihtsa ülesande lahendusi, mistõttu on laias mõttes nagunii tegemist sarnase struktuuriga programmidega. Sellest tulenevalt mõeldakse struktuuri selle töö raames veidi kitsamas kontekstis (pigem ridade tasemel).

2. Programmide lubatud sarnasuse määratlemine

Programmeerimise õpetamisel on oluline kontrollida hindamiseks esitatud tööde vahelist sarnasust. Kuigi selleks on olemas erinevaid automaatseid vahendeid (nt JPlag, MOSS), peab lõpuks siiski inimene otsustama, kas tegu võib olla akadeemilise petturlusega. Selles peatükis käsitletakse petturluse ja programmide lubatud sarnasuse määratlemise keerukust ja võimalikke definitsioone, tutvustatakse õpetajate ja õppijate perspektiivi eelnevale probleemile ning tuuakse välja peamised murekohad.

2.1 Defineerimise keerukus

Akadeemilise petturluse täpne defineerimine on keeruline, kuna lubatud käitumise piiritlemine on komplitseeritud ning sõltub sageli spetsiifilisest situatsioonist. Dick *et al.* (2002) on seetõttu pakkunud välja praktilise definitsiooni, mida peab institutsioonipõhiselt täpsustama. Nende sõnul on tegemist petturlusega, kui saab vastata jaatavalt kahele järgnevale küsimusele:

1) kas käitumine rikub hinnatavale ülesandele püstitatud nõudeid;
2) kas käitumine rikub institutsiooni poolt kehtestatud standardeid?

Lisaks sellele tuleb võtta arvesse käitumise tõsidust (nt kui oluline oli ülesanne kogu kursuse raames ning kas tegu toob kellelegi otsest kahju).

Tartu Ülikooli õppekorralduseeskiri defineerib akadeemilist petturlust järgnevalt: “Akadeemiline petturlus on:

1) õpiväljundite hindamisel lubamatute materjalide kasutamine;
2) õpiväljundite hindamisel teadmiste omavaheline lubamatu vahetamine;
3) teise üliõpilase eest hindamisel osalemine;
4) teise töö esitamine enda nime all või selle osade kasutamine ilma nõuetekohase viitamiseta (plagieerimine);
5) iseenda töö uuesti esitamine samas või teises õppeaines, v.a juhul kui õppejõud on seda lubanud.”

Selles töös käsitletakse peamiselt punktidega 2 ja 4 seonduvaid probleeme.

Programmide sarnasuse uurimisel teeb olukorra keeruliseks see, et võrreldes tekstiliste ülesannetega (nt esseed), on võimalike lahenduste hulk oluliselt väiksem, kuna see on

limiteeritud programmeerimiskeele süntaksiga. Eriti keeruline on olukorda hinnata lihtsamate ülesannete puhul, kuna nende puhul kontrollitakse tavaliselt spetsiifilisi oskusi, seega on loomingulisuse jaoks ruumi vähem, lisaks pole tingimata tekkinud veel isikupärast stiili.

Programmide sarnasust on püütud liigitada ka erinevateks tüüpideks. Walenstein *et al.* (2007) on eristanud süntaktilist (ingl *syntactic*) ehk esinduspõhist (ingl *representational*) ja semantilist (ingl *semantic*) ehk käitumispõhist (ingl *behavioral*) sarnasust. Süntaktilise sarnasuse kontrollimisel lähtutakse koodist kui sümbolite jadast ning vaadatakse teksti sarnasust (nt Levenshteini kaugustega). Semantilise sarnasuse puhul võivad sarnasused esineda näiteks kasutatud funktsioonides. Kuna selles töös analüüsitakse küllaltki lihtsa ülesande lahendusi, siis taolist eristust ei tehta.

Programmide sarnasuse võimalikeks põhjusteks on toodud muuhulgas järgmisi aspekte: õppijad järgivad neilt nõutud stiili/juhiseid; õppijad on oskuste poolest sarnasel tasemel; neil on olnud näidisteks ühesugused ülesanded; võimalike funktsioonide ja lahenduste hulk on sageli küllaltki limiteeritud; lahendusi käsitleti osaliselt tundides; lahendus põhineb eelneval ülesandel; õpitakse sama õpiku põhjal; tehti lubatud määral koostööd (nt võimalike lahendusviiside arutamine); õppijad jagasid ülesande omavahel osadeks ja panid pärast tulemuse kokku; väline osapool lahendas ülesande tasu eest ära; ühistööna tehtud lahendus esitati eraldi; kood kopeeriti mujalt (jälgi võib olla peidetud nt kommentaaride lisamise/muutmisega, muutujate nimede ja struktuuri muutmisega) (Mann & Frew, 2006).

2.2 Lähtekoodi plagiaadi mõiste

Kuna programmide sarnasuse võimalikke põhjuseid on palju, on selguse huvides oluline selgitada plagiaadi mõistet küllaltki täpselt. Cosma ja Joy (2008) on oma töös küsitlenud Suurbritannias programmeerimist õpetavaid õppejõude ja palunud neil hinnata erinevaid hüpoteetilisi situatsioone. Saadud tulemuste põhjal pakuti lähtekoodi plagiaadi määratlemiseks järgnevat definitsiooni. „Lähtekoodi plagiaat võib esineda programmeerimisülesande puhul juhul, kui õppija kasutab kellegi teise lähtekoodi ning ei viita vastavalt nõuetele, kas teadlikult või teadmatusest tulenevalt, esitades seda seega enda tööna.“ Selle seletuse kohaselt koosneb plagiaat kolmest osast: lähtekoodi omandamine (olgu see esialgse autori loaga või ilma), saadud koodi uuesti kasutamine teises lahenduses ning korrektse viitamise puudumine. Kusjuures ülesande all mõeldakse sellist, mida arvestatakse kursuse hinde panemisel. Järgnevalt on toodud

tolles artiklis pakutud selgitused, milliste situatsioonide puhul on tegemist lähtekoodi omandamise, uuesti kasutamise ning ebakorrektse viitamisega.

Uuesti kasutamise alla kuuluvad järgnevad juhud:

1) lähtekoodi kopeerimine ilma selles muudatusi tegemata;
2) lähtekoodi kopeerimine ja selles minimaalsete või mõõdukate muudatuste tegemine (kus tulemus sisaldab endiselt fragmente esialgsest koodist);
3) võõra lähtekoodi täielikult või osaliselt teise keelde "tõlkimise" puhul võib olla tegemist plagiaadiga (see sõltub keelte omavahelisest sarnasusest ja "tõlkimisele" kulunud töö määrast; on võimalik, et tegemist ei ole plagiaadiga, kui õppija on saanud ideid teises keeles kirjutatud koodist);
4) lähtekoodi automaatse vahendiga genereerimine (juhul, kui sellise vahendi kasutamine pole ülesande kirjelduses selgelt lubatud).

Juhul, kui võõra (või enda varasemalt esitatud lahenduses kasutatud) lähtekoodi kasutamine pole lubatud, kuid seda on tehtud koos viidetega, võib tegemist olla pigem ülesandele püstitatud nõuete rikkumise kui plagiaadiga.

Originaali autori loaga või loata lähtekoodi omandamise alla kuuluvad järgnevad juhud:

1) teisele isikule koodi kirjutamise eest maksmine;
2) teise õppija lähtekoodi varastamine;
3) ühe või enama õppijaga koos töötamine ning selle tulemusena sarnaste lahenduste esitamine, juhul, kui ülesanne nõudis individuaalset tööd;
4) teises grupis, kuid sama ülesannet lahendavate õppijaga koodifragmentide vahetamine (olenemata sellest, kas see toimub ülejäänud grupiliikmete nõusolekuga või mitte).

Ebakorrektse viitamise alla kuuluvad järgnevad juhud:

1) lähtekoodi allikale ja autorile oma lahenduses ja dokumentatsioonis mitteviitamine (tekstilise viitena kommentaari näol);
2) võltsitud viidete kasutamine (viited, mille õppija mõtles ise välja, viidatavat tegelikkuses ei eksisteeri), tegemist on fabritseerimisega, mis on ühe akadeemilise petturluse vorme;
3) valede viidete kasutamine (viidatav eksisteerib, kuid ei vasta kopeeritud koodile), tegemist on ühe akadeemilise petturluse vormiga.

Lisaks tuuakse selle punkti all välja programmi väljundi muutmine, eesmärgiga jätta muljet, justkui programm töötaks korrektselt, kuigi tegelikkuses on programm vigane (tegemist on ühe akadeemilise petturluse vormiga).

Culwin *et al.* (2001) toovad välja, et Suurbritannia kõrgkoolide küsitlemise käigus saadud tulemused viitavad sellele, et algajate kursustel on plagiaadi osakaal suurem kui edasijõudnute kursustel. Selle võimalikeks põhjusteks peeti seda, et petturid on jäänud alguses vahele; plagieerivad nõrgemate oskustega õppijad, kes ei jätka edasijõudnute kursustel; või on kursuse tempo selline, et otsustatakse plagieerida, kuid kursuse käigus saavutatakse piisavad oskused, et järgmistel kursustel hakkama saada.

2.3 Õppejõudude perspektiiv

Programmeerimise õpetaja töö üheks osaks on näiteks kodutööde puhul kontrollida, kas õppija on kasutanud akadeemilist petturlust. Äärmuslike juhtumite korral, kus kaks suuremamahulist tööd on omavahel identsed, võib plagiaati kahtlustada küllaltki suure kindlusega. Kuid keerulisemate juhtumite korral esineb ka õppejõudude vahel erimeelsusi selles, mis kategoriseerub plagiaadi alla ning mis mitte.

Cosma ja Joy (2008) küsitlesid lähtekoodi plagiaadi mõiste defineerimiseks Suurbritannia programmeerimise õppejõude. Selle käigus paluti hinnata nelja stsenaariumi puhul, kas tegemist on plagiaadiga või muul moel akadeemilise petturlusega. Tulemustest selgus et küllaltki üksmeelselt arvati, et võõra koodi kopeerimise puhul ilma selles muudatuste tegemiseta ega viitamiseta on tegemist plagiaadiga. Siinkohal toodi välja ka seda, et programmeerimisülesannete puhul ei pruugi plagiaat piirduda vaid lähtekoodi kopeerimisega, see võib sisaldada ka kommentaaride, sisendandmete ja kasutajaliidese disaini kopeerimist. Probleeme tekitas aga võõra koodi kasutamine koos sellele viitamisega. Selle osas kommenteeriti, et kuna näiteks objektorienteeritud keskkondade puhul julgustatakse kasutama olemasolevaid materjale, ei tähenda kopeeritud kood veel seda, et tegemist oleks plagiaadiga. Oluliseks peeti seda, et on selgelt väljendatud lubatud piire ja nõudeid. Võõra olemasoleva koodi ühest keelest teise tõlkimise ja allikale mitte viitamise puhul peeti oluliseks hinnata nähtud vaeva. Automaatsete vahendite kasutamisel võib tegemist olla plagiaadiga, kuid kui kood tõlgitakse tervenisti iseseisvalt, on tõenäoline, et tegemist pole plagiaadiga.

Kõige suuremad erisused esinesid eneseplagiaadi hindamise puhul, ehk oma varem esitatud töö kasutamisel ilma sellele viitamata. Suurem osa õppejõududest pidas sellist käitumist

akadeemiliseks rikkumiseks, kellest omakorda ligi kolmandik pidas seda plagiaadiks. Kuid leidus ka õppejõudusid, kelle hinnangul ei ole sobilik piirata teise ülesande lahenduseks koostatud koodi uuesti kasutamist, kui üldiselt julgustatakse olemasolevate materjalide kasutamist. Joy *et al.* (2010) õppijate arusaamadele keskendunud töös oli õppejõudude hinnangul aga varem esitatud töö viitamata kasutamise puhul tegemist selge plagiaadijuhtumiga. Õppijate omavahelise koostöö osas oldi üksmeelel, et ideede jagamine ja ülesannete üle arutlemine on väärtuslik, kuniks ei kopeerita üksteise lahendusi.

2.4 Õppijate perspektiiv

Selleks, et akadeemilist petturlust vältida, on esmalt oluline seda mõista. Selgub aga, et programmeerimise õppijate vahel esineb suuri erinevusi plagiaadi määratlemise osas. Lisaks sellele võib õppijat arusaam erineda küllaltki tugevalt õppejõudude nägemusest.

Joy *et al.* (2010) on oma töös uurinud, milline on õppijate nägemus plagiaadist programmeerimise puhul. Nad palusid küsitluse vahendusel hinnata kuue erineva teema osas, kas tegemist võiks olla plagiaadiga. Küsitlusele vastas 770 õppijat Suurbritannias, kellest 98,3% nõustusid väitega, et nad tunnevad, et mõistavad plagiaadi olemust. Tulemustest selgus, et suurem osa õppijatest (89,9%) hindas õigesti, et tegemist pole plagiaadiga, kui kasutatakse oma varasemalt esitatud koodi ning sellele ka viidatakse, kuid vaid 6,8% vastas, et tegemist on plagiaadiga, kui lahenduses kasutatakse oma varem esitatud koodi ning sellele ei viidata, ometigi oli selles küsitluses kasutatud õppejõudude hinnangul siinkohal tegemist selge plagiaadijuhtumiga. Viitamine tekitas segadust ka valede viidete kasutamisel (isegi kui seda on tehtud tahtmatult), mida ei pidanud plagiaadiks 53,5% õppijatest, ning võõra koodi teise keelde tõlkimise korral ilma allikale viitamata, mida pidas õigesti plagiaadiks vaid 49% vastanutest.

Lisaks sellele ilmnes, et õppijatel on keeruline tõmmata piiri võõra koodi kasutamise ja sellest inspiratsiooni saamise vahel. Raskusi tekitab ka individuaalse töö puhul teistega koos lubatud lahendamise määr. Näiteks kui individuaalse töö puhul lahendatakse ülesannet kahekesi koos ja tulemusena esitatakse sarnased lahendused, oli 62% vastanute hinnangul tegemist plagiaadiga. Ka juhul, kui grupitöö puhul jagavad kahte erinevasse gruppi kuuluvad õppijad omavahel fragmente koodidest, olid õppejõudude hinnanguga kooskõlas 62% õppijatest, kes pidasid seda plagiaadiks. Õigesti saadi aru siiski sellest, et juhul kui programmi väljundit muudetakse, eesmärgiga jätta mulje programmi korrektsest tööst (kui tegelik programm oli

vigane), siis pole tegemist plagiaadiga (pigem on tegemist teise akadeemilise petturluse viisi, tulemuste võltsimisega).

Sheard *et al.* uurisid oma 2003. aasta töös IT tudengite suhtumist ning nende endi kogemust akadeemilise petturlusega. Tulemustest selgus, et 80% küsitletutest oli praktiseerinud vähemalt ühte 18 etteantud stsenaariumist, küllaltki aktsepteeritavaks peeti õpikust või Internetist essee tarbeks teksti kopeerimist ning juhtu, kus harjutusülesandele, mis moodustab hindest 5%, annab klassikaaslane vastuse ette, juhul kui enda kasutatava arvutiga esines probleeme. Lisaks leiti, et peamised välised faktorid, mis põhjustavad petturlust, on ülesande halb ülesehitus, materjalide vähesus ning probleemid tarkvara ja muude vahenditega. Aasheim *et al.* (2012) näitasid ka oma töös, kuidas plagiaadi teema käsitlemise ning selgemate piiride määratlemisega on õppijate tunnetust plagiaadist võimalik muuta. Seda kinnitas ka Mason *et al.* (2019) artikkel, kus leiti, et õppijate selgem informeerimine vähendas oluliselt plagiaadijuhtumeid.

2.5 Kokkuvõte

Kirjanduse põhjal saame öelda, et programmide sarnasuse lubatavuse hindamise teeb keerukaks see, et võimalike lahenduste hulk on limiteeritud, õpitakse samade materjalide põhjal, segadust tekitab ka lubatud koostöö piiritlemine ja olemasoleva koodi kasutamine. Seega võivad sarnasust põhjustada nii pahatahtlik tegevus, teadmatusest tulenev tingimuste rikkumine (nt lubamatu koostöö näol) kui ka muud tingimused, mis viivad paratamatult küllaltki sarnaste töödeni (nt lihtsad ülesanded, kus on detailselt selgitatud nõutud lahendusviis).

Selle töö järgnevates osades analüüsitakse programmide sarnasust nii käsitsi kui ka automaatkontrolli vahendit kasutades. Uuritakse, kuivõrd on ühe kursuse mentorid üksmeelel erinevate lahenduste paaride sarnasuse hindamisel. Lisaks vaadatakse, kuidas selgitavad mentorid programmide sarnasust ning kui keeruliseks seda ülesannet algajate kursuse puhul peetakse.

3. Ühe ülesande lahenduste sarnasuse autoripoolne määratlemine

Töös analüüsitakse Tartu Ülikooli programmeerimiskursuse “Tehnoloogia tarbijast loojaks” raames esitatud ühe ülesande 24 lahendust. Töid analüüsiti kolmes faasis: 1) uuriti kõiki lahendusi neid käsitsi läbi vaadates; 2) vaadati Moodle’i õpikeskkonnas pakutava sarnasuse analüüsi tulemusi ja 3) valiti eelneva kahe sammu tulemuste põhjal välja kuus paari töid, kusjuures ühte lahendust modifitseeriti. Neid paare paluti küsitluse vahendusel hinnata kursuse mentoritel. Paaride valiku põhimõtteid ja küsitluse tulemusi käsitletakse töö neljandas peatükis. Selles peatükis selgitatakse kursuse ja ülesande valikut, tutvustatakse käsitsi analüüsimise metoodikat ja tulemusi, Moodle’i analüüsi tulemusi ning võrreldakse kahte viimast omavahel.

3.1 Kursuse ja ülesande valik

Uuritav lahendus valiti kursuselt “Tehnoloogia tarbijast loojaks”. Tegemist on 10 nädala pikkuse e-kursusega, mis on mõeldud 16–26-aastastele noortele, kellel varasem kokkupuude programmeerimisega kas puudub või on vähene. Õppijate toetamiseks ja lahenduste hindamiseks on kursusel mentorid. Koolides võib kursus toimuda ka osaliselt auditoorselt. Kursus valiti, kuna see oli lõputöö kirjutamise ajal käimas, seega oli olemas ligipääs hiljuti esitatud lahendustele. Lisaks on sellel kursusel õppuritele toeks mentorid, kes esitatud töid hindavad ja nende sarnasusi Moodle’i abiga peavad kontrollima, mis andis võimaluse paluda neil teatud lahenduste sarnasust kommenteerida.

Kursuselt valiti välja 5. nädala ülesanne *Jukebox*, kuna selleks nädalaks oli läbitud juba küllalt palju teemasid, seega nõudis ülesande lahendamine juba veidi rohkem oskusi ja koosnes mitmest elemendist. Esimeste nädalate tööde puhul oleks võimalike lahenduste hulk liiga limiteeritud, kuna ülesanded on väga lihtsad ja lühikesed. Selle ülesande puhul leiti, et võimalike lahenduste hulk võiks olla piisavalt suur ja ülesanne selline, et lubab kasutada piisavalt loovust ja näidata oma stiili. Lahendusi käsitletakse anonümiseeritud kujul, kus lahenduse nime moodustab täht “o” ja sellele antud järjekorranumber.

Ülesande *Jukebox* kirjeldus on järgnev:

> *Ada tahab valida plaadiautomaadist laulu ja uurib, milliseid laule masin mängib. Muusikapalad on kirjas failis, kus iga laul on eraldi real.*
>
> *Koostada programm, mis*
>
> *- küsib kasutajalt failinime (kasutaja sisestab failinime koos laiendiga, nt jukebox.txt);*
>
> *- loeb sisestatud nimega failist andmed;*
>
> *- näitab kõiki laule koos järjekorranumbritega (alates 1);*
>
> *- küsib kasutajalt, mitmendat laulu ta soovib (kasutaja sisestab alati täisarvu);*
>
> *- väljastab ekraanile vastavalt valitud arvule muusikapala.*

3.2 Käsitsi analüüsimine

Selles peatükis selgitatakse käsitsi analüüsimise metoodikat ja tulemusi. Töö käigus vaadati läbi kõik 24 lahendust.

3.2.1 Käsitsi analüüsimise metoodika

Lahenduste sarnasuse määratlemist alustati tööde käsitsi läbi vaatamisega, eesmärgiga neid kategoriseerida. Tööde uurimisel otsiti kõigepealt lahendusi, mis teistest silmapaistvalt erinesid ning jäeti need edasisest uurimisest välja. Siin eraldati lahendused, mis ei kompileeru või on selgelt valed, leidus ka üks lahendus, mis põhines selgelt teisel ülesandel. Kokku langes välja kuus tööd. Ülejäänud 18 lahenduse uurimisel kujunes välja kolm kategooriat, kuhu kõik tööd paigutati.

Lahenduste kategoriseerimisel lähtuti peamiselt lahenduse struktuurist (peamiselt ridade tasemel). Peamiselt vaadati failist info sisse lugemise viisi ja tsüklite ülesehitust. Lisaks sellele vaadati muutujanimesid ja funktsioonide kasutust.

3.2.2 Käsitsi analüüsimise tulemused

Esimesse kategooriasse kuulus 12 tööd (o1, o2, o3, o4, o6, o8, o10, o11, o12, o13, o16, o24). Nende puhul oli ühtne failist lugemise viis ja väga sarnane või identne programmi struktuur. Siinkohal on siiski oluline märkida, et kuna tegemist on algajate kursusega, on ka suurem tõenäosus, et faili lugemiseks on kopeeritud otse sama õpiku näide ning seega on ka sarnasused

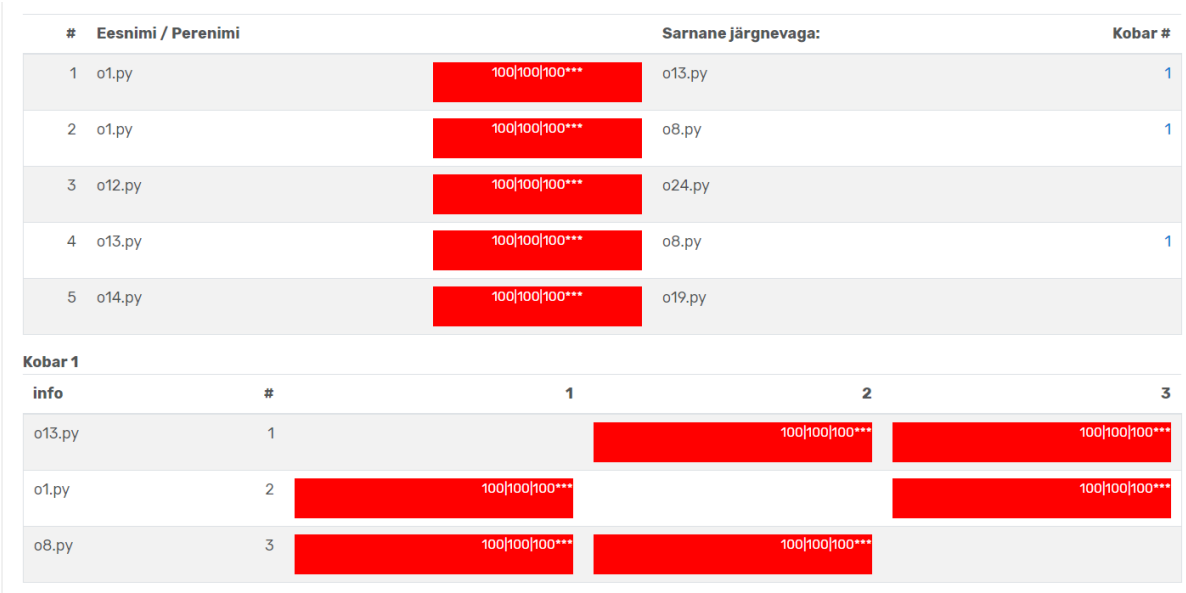
suuremad. Lisaks oli näha lahendusi, mille puhul ainsaks erinevuseks oli muutujanimi (ka see erinevus võis olla väga väike, näiteks laulude listi nimeks “muusika” vs “musa“), kusjuures esines ka muutujate nimesid, mille puhul ei esine seost muutuja sisuga (nt failinimi “q”, loendur “w”, laulu järjekorranumber “e”).

Teise kategooriasse kuulus 4 tööd (o14, o19, o23, o9). Nende puhul on ühesugune failist reakaupa sisendi lugemine, kuid ka kogu töö struktuur on identne, erinevused on vaid muutujate nimedes, mõne töö puhul on kasutatud rohkem tühikuid ja tühje ridu.

Kolmandasse kategooriasse kuulus 2 tööd (o22, o7). Need kaks lahendust tundusid autorile sarnased oma struktuuri ja kasutatud lahendusviiside poolest (eriti jäi silma mõlemas lahenduses kasutatud ebatavaline failist lugemise viis), samas leidus selliseid erinevusi, mille põhjal neid automaatselt kokku ei grupeeritaks. Seega tundus, et nende kahe töö puhul oleks võimalik vaadata, kuidas on võimalik plagiaadi puhul oma jälgi peita. Selle tulemusena on võimalik taolisi märke töödest otsida. Seda paari paluti hinnata ka mentoritel, tulemused on peatükis 4.5.

3.3 Moodle’i analüüsi tulemused

Moodle’i õpikeskkonnas on võimalik programmide sarnasust automaatselt hinnata. Selleks saab üles laadida lahendusi sisaldava ZIP-faili. Lahendustest moodustatakse kõikvõimalikud paarid ning tulemusena näidatakse kõigepealt paare üksteise all loeteluna (koos nendevahelist sarnasust väljendava hinnanguga) ning seejärel omavahel sarnaste lahenduste klastritena (joonis 1). Seejuures saab ka valida, kui palju tulemusi maksimaalselt väljastatakse (ehk kui mitu esimest tulemust väljastatakse). Automaatsel hindamisel jäetakse välja kommentaarid ja erinevusteks ei arvestata muutujanimede erisusi (Wanhenheim *et al.*, 2015). Sarnasuse analüüsi vahendit rakendati kõigile 24 lahendusele, seega vaadeldi ka neid lahendusi, mis jäeti välja käsitsi analüüsi käigus.



Joonis 1. Esimese viie tulemuse väljastamisel saadav vaade.

Programmidevahelist sarnasust väljendatakse punastes ristkülikutes kolmest arvust koosneva hinnanguga. Need arvud väljendavad erinevaid mõõtmistulemusi, täpsemaid selgitusi võib lugeda Wanhenheim *et al.* teosest "*Developing Programming Courses with Moodle and VPL-The Teacher's Guide to the Virtual Programming Lab*". Automaatkontrolli kasutaja jaoks piisab teadmisest, et suuremad arvud väljendavad suuremat sarnasust.

Siinjuures on siiski oluline märkida, et ei saa olla kindel, et teatud arvust suuremate hinnangutega tööd on kindlasti plagiaadid. Keerukamate lahenduste korral võib sarnasuse määr olla madalam, lihtsamate lahenduste puhul on sarnasused suuremad. Siit on näha see, et ka masinliku sarnasuse kontrolli tulemuste tõlgendamine ja nende põhjal järelduste tegemine võib olla keeruline. Kuna vaid arvulise hinnangu põhjal ei saa kindlaid järeldusi teha, on oluline, et inimene programmi poolt esile tõstetud tööd ise üle vaataks.

o1.py

```
1  a=str(input())
2  fail=open(a)
3  m=1
4  muusika=[]
5  print("Muusikapalade valik:")
6  for i in (fail):
7      muusika.append(i)
8      print(str(m)+".",i)
9      m=m+1
10 g=int(input("Valige laulu järjekorranumber: "))
11 g=g-1
12 print("Mängitav muusikapala on "+ muusika[g])
```

o13.py

```
===  1  a=str(input())
###  2  file=open(a)
===  3  m=1
###  4  musa=[]
===  5  print("Muusikapalade valik:")
###  6  for i in (file):
###  7      musa.append(i)
===  8      print(str(m)+".",i)
===  9      m=m+1
### 10  b=int(input("Valige laulu järjekorranumber: "))
### 11  b=b-1
### 12  print("Mängitav muusikapala on "+ musa[b])
```

Joonis 2. Kahe sarnase programmi võrdlus.

Kõiki väljatoodud paare on võimalik ka täpsemalt vaadata saadud hinnangul klikkides, programm toob ka rea kaupa välja, millised read on identsed (tähistusega "===") ning millistes

esineb erinevusi (tähistusega “###”). Seda on demonstreeritud joonisel X2 lahendustega o1 ja o13, mida automaatkontroll pidas väga sarnasteks.

Joonis 3. Moodle’is 15 tulemuse väljastamisel saadavad kobarad.

Kuna esimese viie paari kuvamisest ei piisanud, vaadati esimese 15 tulemuse väljastamisel saadavaid tulemusi. Joonisel 3 on kujutatud selle käigus tuvastatud kobaraid. Teine kobar langeb täpselt kokku käsitsi uurimisel tuvastatud teise kobaraga.

3.4 Tulemuste võrdlus

Üldiselt võib öelda, et Moodle’is sarnasuste kontrollimine on kasulik, kuna see toob esile väga sarnased paarid, mille puhul inimene saab neid ise lähemalt uurida ja otsustada, kas tegemist võib olla plagiaadiga. Saadavate tulemuste tõlgendamine võib olla keeruline, eriti kui on vaja otsustada, milliseid paare peaks käsitsi läbi vaatama ning milliseid mitte.

Moodle’i tuvastatud kahest kobarast teine langes täpselt kokku ka käsitsi analüüsi käigus kujunenud teise kobaraga. Käsitsi analüüsi esimene kobar sisaldas endas kõiki Moodle’i analüüsi esimesse kobarasse paigutatud lahendusi. Automaatse sarnasuse tuvastuse probleem tuleb välja aga käsitsi analüüsi kolmanda kobara puhul, kuhu kuulus paar o7 ja o22. Paar tekitas autoris kahtlusi (peamiselt ebatavalise failist andmete lugemise viisi tõttu), kuid lahendused on siiski piisavalt erinevad, et automaatne tuvastus nendes vähemalt suurt kahtlust ei näinud. Et selles paaris selgusele jõuda, paluti seda hinnata küsitluse käigus ka mentoritel, tulemused on peatükis 4.5. Järgnevas peatükis käsitletaksegi mentoritele suunatud küsitluse käiku ja selle tulemusi.

4. Ühe ülesande lahenduste sarnasuse mentorite poolne määratlemine

Ülesande lahenduste sarnasusi uuriti kõigepealt käsitsi ja seejärel Moodle'is. Saadud tulemuste põhjal valiti välja kuus paari, mille osas küsiti hinnangut kursuse "Tehnoloogia tarbijast loojaks" mentoritelt. Mentorite eksperthinnangut otsustati küsida seetõttu, et nad juba tegelevad oma töös sarnase hindamisega. Lisaks on neil juba eelnev teadmine ülesannete sisust, mistõttu ei nõudnud vastamine üleliia suurt ajakulu. Järgnevalt on toodud mentorite ja paaride valiku selgitused, selgitatud küsitluse metoodikat ning näidatud küsitluse tulemusi esmalt iga paari kohta eraldi ning lõpuks iga küsimuse osas üldiselt.

4.1 Küsitluse metoodika

Küsitlus koosnes kuuest lahenduste paarist, mille puhul paluti hinnata, kas tegemist võiks olla plagiaadiga. Seda jagati veebi teel kõigile kursuse mentoritele 2020. aasta märtsi lõpus, vastamiseks anti kaks nädalat. Kursuse 33 mentorist vastas küsitlusele 11.

Iga programmipaari puhul paluti mentoril hinnata seitsmepalliskaalal, kuivõrd tuntakse, et tegemist on plagiaadiga, tuua argumente plagiaadi poolt ja vastu ning selgitada, mida võis olla tehtud jälgede segamiseks. Küsitluse lõpus paluti mentoritel hinnata ka seda, kui lihtne oli küsimustele vastata ning millised olid keerulisemad olukorrad. Töös on tulemusi esitatud nõnda, et küsitluses esimesena küsitud hinnang on toodud iga paari puhul viimasena, kuna see võimaldab võtta kokku eelnevalt välja toodud aspekte. Selguse huvides on läbivalt kaudse kõneviisi asemel ka vajadusel selgitatud, millal on tegemist autori hinnanguga ning millal mentorite hinnanguga.

4.2 Küsitluse paaride valimine

Küsitluse paaride valimisel lähtuti käsitsi tehtud analüüsi tulemustest. Paarid moodustati nõnda, et sarnasuste põhjused oleksid võimalikult erinevad, eelkõige vaadati muutujanimesid ja üldist struktuuri. Iga paari omadus ja seega ka valiku põhjus kajastub iga paari analüüsi pealkirja nimes, täpsem tutvustus on toodud iga paari analüüsi alguses. Saadud paarid on järgnevad:

- sarnaste muutujanimede, kuid erineva struktuuriga paar;
- sarnaste muutujanimede ja struktuuriga paar;
- identsete muutujanimede ja erineva struktuuriga paar;

- osalt erinevate muutujanimede ja osalt identsete ridadega paar;
- sarnaste muutujanimede ja identse struktuuriga paar;
- erinevate muutujanimede ja struktuuriga paar.

Ühel juhul otsustati lahendust modifitseerida, kuna sooviti lisada paari, mille puhul on koodi struktuur väga erinev ja muutujate nimed identsed, kuid sellist näidet lahenduste hulgas ei leidunud. Modifitseeritud lahendusele anti nimeks o25. Ülejäänud juhtudel on kasutatud lahendusi originaalsetel kujudel.

4.3 Sarnaste muutujanimede, kuid erineva struktuuriga paar

o15.py

```
a = input("Palun sisestage failinimi: ")
fail = open(a, encoding="UTF-8")
print("Muusikapalade valik: ")
laul = []
laulud = []
b = 0
f = 1
g = 0

for rida in fail:
    laul = rida
    laulud = laulud + [rida.strip("\n")]
    print(str(f)+ '. ' + laul)
    f += 1
fail.close()

print('')
h = int(input('Valige laulu järjekorranumber: '))
h -= 1
print('Mängitav muusikapala on ' + laulud[h] + '.')
```

o6.py

```
a = str(input())
fail = open(a)
b = 1
muusika = []
print('Muusikapalade valik:')
for i in (fail):
    muusika.append(i)
    print(str(b) + '.', i)
    b = b + 1
g = int(input('Valige laulu järjekorranumber: '))
g = g - 1
print('Mängitav muusikapala on ' + muusika[g])
```

Joonis 4. Lahendused o15 ja o6.

Lahenduste o15 ja o6 (joonisel 4) võrdlemisel leidis autor, et kasutatud muutujate nimed on sarnased ning lisaks sellele ka küllaltki ebaloogilised, kuna ei väljenda muutuja sisu. Nimedeks oli kasutatud üksikuid tähti, mis osaliselt ka kattusid. Mõlemal juhul justkui oleks kasutatud tähestiku algusest järjestikuseid tähti, kuid osad on vahelt puudu. See võib olla põhjustatud sellest, et lahendamise käigus kasutati teisi muutujaid, mida lõpplahenduses siiski vaja ei läinud ja ära kustutati. Struktuuri osas näivad lahendused veidi erinevad, peamiselt muutujate arvu ja ridade paigutuse tõttu. Lisaks on erinev muutuja väärtuse ühe võrra suurendamine/vähendamine (`f += 1` vs `b = b + 1`). Sarnastes osades kasutatakse standardseid lahendusviise, kuid mõlemad sisaldavad muutujaid, mida tegelikult ei kasutata. Parempoolses koodis on kasutatud ka üleliigseid funktsioone (esimesel real on rakendatud kasutajalt saadud sisendile asjatult funktsiooni `str()`, for-tsükli puhul on lisatud ebavajalikud sulud). Nendel põhjustel jõuti arvamusele, et pigem pole tegemist plagiaadiga, kuid on võimalik, et lahendusi on omavahel arutatud. Järgnevalt on toodud mentorite hinnangud käesolevale paarile.

4.3.1 Argumendid plagiaadi poolt

Küsitluses paluti selgitada, milliseid argumente saab esitada selle väite poolt, et tegemist on plagiaadiga. Siinkohal toodi välja järgnevaid aspekte.

- Kahtlust äratab see, et vasakpoolses koodis on kasutatud ühel korral jutumärke, teisel korral ülakomasid.
- Parempoolses lahenduses on kasutusel muutujad "b" ja "g", vasakpoolsel lahendusel on samanimelised muutujad olemas, kuid neid kordagi ei kasutata (tekitab kahtlust, et vasakpoolse programmi autor lisas need seetõttu, et nägi kedagi teist neid kasutamas).
- Ekraanile väljastatav tekst on liiga sarnane (kuid siinkohal leidus ka mentor, kes tõi välja, et see võib tuleneda etteantud programmi töö näitest).
- Muutujate nimed on sarnased.
- Kasutajalt saadud sisendi ja faili nime muutujate nimed on ühesugused (vastavalt "a" ja "fail", ridadelt 1 ja 2).
- Leidus omavahel sarnaseid ridu, mille puhul suurem erinevus on vaid muutujate nimedes (lahenduse o15 read 18–20 ja lahenduse o6 read 10–12, kus küsitakse kasutajalt järjekorranumbrit, vähendatakse muutuja väärtust ühe võrra ja väljastatakse vastus, ühel juhul on väljastatavasse lisatud ka lõppu punkt).

Selgitustes mainiti, et vasakpoolne justkui kirjutaks parempoolset lahendust lahti, püüdlikult esitlust muutes. Üheks võimalikuks selgituseks pakuti ka seda, et võib-olla prooviti naabrilt piiluda, kuid siis loobuti, kuna saadi ise aru. Leiti ka, et on aimata, et on tehtud väike katse kopeerimiseks, kuid siis sellest loobutud ning edasi ülesannet iseseisvalt lahendatud. Kaks mentorit vastasid siinkohal vaid, et ei pea töid plagiaadiks.

4.3.2 Argumendid plagiaadi vastu

Küsitluses paluti selgitada, milliseid argumente saab esitada selle väite poolt, et tegemist ei ole plagiaadiga. Siinkohal toodi välja järgnevaid aspekte.

- For-tsükkel on erinevalt lahendatud (seega, kui oleks tegemist ühe lahenduse ümberkirjutamisega, peaks tegijal olema üpris palju teadmisi sellest, kuidas sama lahendust teisiti esitada).
- Lahenduste "käekiri" on veidi erinev.
- Faili avamisel on ühel juhul kodeering määratud, teisel juhul mitte.
- Järjendisse elementide lisamine on tehtud erinevalt (`append vs +=`).

- Vasakpoolses lahenduses suletakse lõpus fail, parempoolses mitte.
- Oluliselt erinev koodiridade arv.
- Sarnane on vaid väljastatav tekst.
- Kasutajale kuvatava teksti sarnasus tuleneb ülesandes antud programmi töö näitest ja on seega mõistetav.
- Failist andmete lugemise põhimõtted on erinevad.
- Vasakpoolses lahenduses küsitakse kasutajalt täpset sisendit, parempoolses lahenduses ei sõnastata sisendi küsimist.
- Parempoolses lahenduses kasutatakse vähem muutujaid.
- Vasakpoolne lahendus on hästi omapärane, parempoolne on pigem standardne lahendus.

Lisaks kommenteeriti, et kui tegemist peaks olema plagiaadiga, siis on kopeerija näinud nõnda palju vaeva jälgede peitmisega, et sisuliselt poleks siiski enam tegemist plagiaadiga. Toodi välja ka arvamus, et selle ülesande peamine keerukus seisneb järjendisse lugemisel ja see osa on mõlemal lahendatud täiesti erinevalt.

4.3.3 Võimalikud viisid jälgede segamiseks

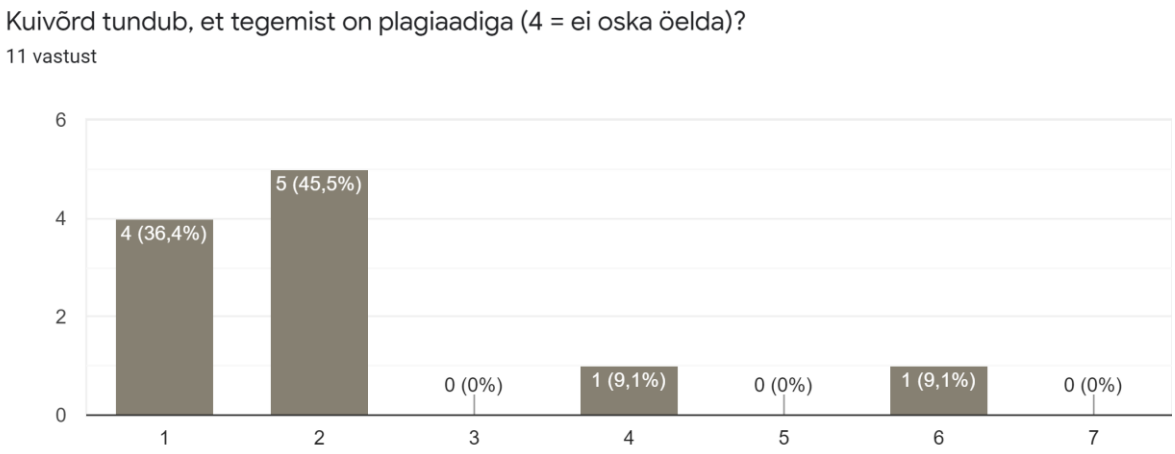
Küsitluses paluti selgitada, milliseid tehnikaid on võidud kasutada jälgede segamiseks. Siinkohal toodi välja järgnevaid aspekte:

- andmete sisse lugemise ja väljastamise käskude ja nende esitamise täpsuse muutmine;
- muutujate nimede vahetamine;
- koodi optimeerimine;
- koodi pikemaks tegemine;
- liitmisel `x = x + 1` asemel `x += 1` kasutamine;
- tühja sõne ekraanile kuvamine;
- ekraanile väljastamise rea paigutuse muutmine;
- `input()` ja `open()` funktsioonide puhul argumentide väljajätmine.

Pakuti välja arvamus, et juhul, kui parempoolse lahenduse puhul on tegemist plagiaadiga, siis pidi mahakirjutaja seda mitmekordselt lihtsustama. Kui aga vastupidi, siis pidi mahakirjutaja tegema palju tööd programmi struktuuri keerulisemaks muutmiseks.

4.3.4 Hinnang sellele, kuivõrd on tegemist plagiaadiga

Mentoritel paluti hinnata seitsmepalliskaalal (kus 1 = kindlasti pole plagiaat, 4 = ei oska öelda, 7 = kindlasti on plagiaat), kuivõrd tundub, et tegemist on plagiaadiga.



Joonis 5. Mentorite hinnang plagiaadi osas, 1 = kindlasti pole plagiaat ja 7 = kindlasti on plagiaat.

Saadud vastuste jaotust illustreerib joonis 5. Kokkuvõtvalt võib öelda, et:

- 9 korral leiti, et ei ole plagiaat (hinne “1” või “2”);
- 1 korral ei osatud öelda (hinne “4”);
- 1 korral leiti, et on plagiaat (hinne “6”).

Seega oli enamuse otsus, et tegemist ei ole plagiaadiga, kuid kuna see polnud üksmeelne, võib järeldada, et lahenduste puhul on aspekte, mis võivad kahtlust äratada. Need tulemused langevad suuresti kokku ka autori hinnangutega (peatüki 4.2 algusest). See isik, kes andis hinde “6”, arvates ainsana, et selle paari puhul võiks pigem olla tegemist plagiaadiga, pidas olulisemateks argumentideks seda, et muutujate nimed on sarnased (“a” ja “b”, “f” ja “g”) ning programmide teised pooled on omavahel sarnased (read 18–20 vasakpoolsel lahendusel ja read 10–12 parempoolsel lahendusel). Teisi mentoreid veenis plagiaadi vastu peamiselt järjendisse lisamise ja faili lugemise erinev lahendusviis ja ka erinev muutujate arv.

4.4 Sarnaste muutujanimede ja struktuuriga paar

o24.py

```
1  x=str(input())
2  fail=open(x)
3  a=1
4  muusika=[]
5  print("Muusikapalade valik: ")
6  for i in (fail):
7    muusika.append(i)
8    print(str(a)+","+i)
9    a=a+1
10 g=int(input("Valige laulu järjekorranumber: "))
11 g=g-1
12 print("Mängitav muusikapala on "+ muusika[g])
```

o16.py

```
1  a=str(input())
2  fail=open(a)
3  m=1
4  muusika=[]
5  print('muusikapalade valik: ')
6  for i in (fail):
7    muusika.append(i)
8    print(str(m)+'.',i)
9    m+=1
10 p=int(input('valige laulu jarjekorranumber: '))
11 p-=1
12 print('mangitav muusikapala on '+muusika[p])
13
14
```

Joonis 6. Lahendused o24 ja o16.

Lahenduste o24 ja o16 võrdlemisel (joonisel 6) leidis autor, et struktuuri ja funktsioonide poolest on tegemist väga sarnaste paaridega. Erinevused on muutujate nimedes (kusjuures mõlemal juhul on need ühetähelised ning lisaks ka ei kajasta muutuja sisu, nt o24 puhul on kasutajalt küsitud failinime tähistatud muutujaga “x”). Kahtlust äratab ka see, et mõlemal juhul küsitakse kasutajalt esimest sisendit ilma küsimust ekraanile printimata, kusjuures saadud sisend teisendatakse ka asjatult sõnadeks (kasutades `str(input())` esimestel ridadel). Suurim erinevus seisneb muutuja ühe võrra suurendamise/vähendamise tehnikas, kuid sellest hoolimata leidis autor, et tegemist võiks olla plagiaadiga. Järgnevalt on toodud mentorite hinnangud käesolevale paarile.

4.4.1 Argumendid plagiaadi poolt

Küsitluses paluti selgitada, milliseid argumente saab esitada selle väite poolt, et tegemist on plagiaadiga. Siinkohal toodi välja järgnevaid aspekte.

- Kõik käsud on samadel ridadel (kõik read täidavad sama funktsiooni).
- Muudetud on vaid pisiasju (muutujate nimed, jutumärkide vs ülakomaga teksti tähistamine).
- Kui inimene on harjunud avaldist ühel viisil kirjutama, siis teeb ta reeglina kõikjal samamoodi, nendes koodides on tõenäoliselt taotletud kunstlikku erinevust loenduri suurendamise avaldise erinevaks muutmisega (read 9 ja 11).
- Real 6 on mõlemal juhul sõna “fail” ümber üleliigsed sulud.
- Koodiridade arv on sama.

Lisaks kommenteeriti, et kuigi on tegemist lihtsa programmiga ja suur sarnasus on ootuspärane, on siiski kummaline, et kõigi ridade sisu on peaaegu identne. See on ka peamine asjaolu, mis mentoreid veenis. Leidus ka neid, kelle hinnangul on tegemist üks-ühele lahendusega.

4.4.2 Argumendid plagiaadi vastu

Küsitluses paluti selgitada, milliseid argumente saab esitada selle väite poolt, et tegemist ei ole plagiaadiga. Siinkohal toodi välja järgnevaid aspekte.

- Kui on õpitud koos ja samade eeskujude järgi, siis jääb siiski mingi võimalus, et ei ole plagiaat.
- Kaheksandatel ridadel on print-käsu puhul kasutatud ühel juhul osade plussmärgi abil liitmist, teisel juhul on viimane komponent lisatud koma abil.
- Erinevad ühetäheliste muutujate nimed.
- Erinevad jutumärgid (ülakoma või jutumärgid).
- Erinev tehnika muutuja ühe võrra suurendamiseks/vähendamiseks (read 9 ja 11).

Siinkohal kommenteeriti, et kuna programm on väike ja lihtne, pole välistatud, et sarnasused on juhuslikud, mis näitab veelkord, et lihtsamate programmide puhul on sarnasuse hindamine võrdlemisi keeruline. Nõrgaks argumendiks peeti ka muutujate nimede erinevust. Üheteistkümnest mentorist neli kommenteerisid, et ei oska tuua mingeid vastuargumente, kuna nende hinnangul on kindlasti tegemist plagiaadiga.

4.4.3 Võimalikud viisid jälgede segamiseks

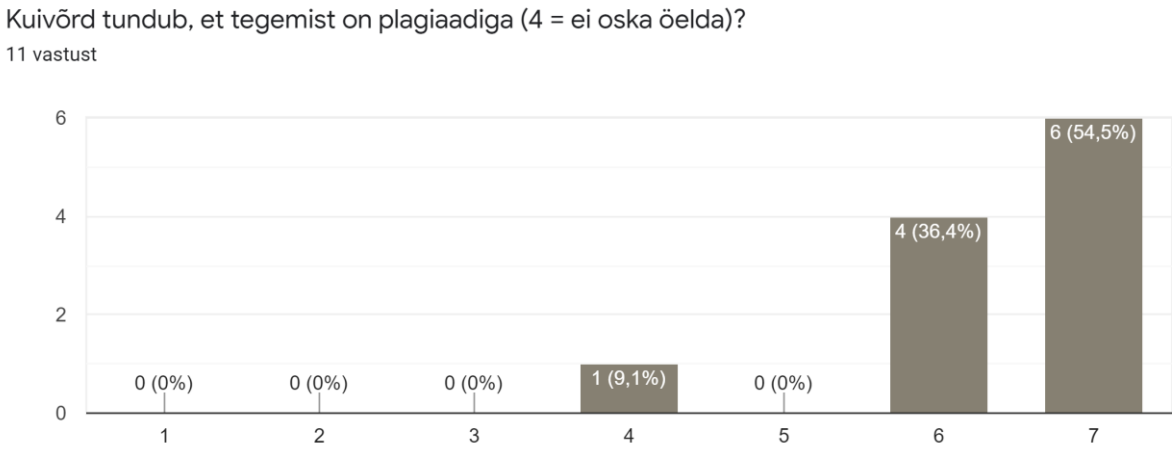
Küsitluses paluti selgitada, milliseid tehnikaid on võidud kasutada jälgede segamiseks. Siinkohal toodi välja järgnevaid aspekte:

- muutujate nimede muutmine;
- teksti esitamise viis (jutumärkide või ülakomadega);
- muutuja väärtuse ühe võrra suurendamise/vähendamise tehnika muutmine;
- suure/väikse algustähe muutmine;
- täpitähtede asendamine/lisamine;
- print-lauses komaga vs plussmärgiga liitmine.

Programmidevahelisi erinevusi selgitati sellega, et tehtud on “kosmeetilisi pisimuudatusi”. Suuremaks jälgede segamise viisiks peeti muutuja väärtuse ühe võrra muutmise tehnika vahetamist ridadel 9 ja 11.

4.4.4 Hinnang sellele, kuivõrd on tegemist plagiaadiga

Mentoritel paluti hinnata seitsmepalliskaalal (kus 1 = kindlasti pole plagiaat, 4 = ei oska öelda, 7 = kindlasti on plagiaat), kuivõrd tundub, et tegemist on plagiaadiga.



Joonis 7. Mentorite hinnang plagiaadi osas, 1 = kindlasti pole plagiaat ja 7 = kindlasti on plagiaat.

Saadud vastuste jaotust illustreerib joonis 7. Kokkuvõtvalt võib öelda, et:

- 1 korral leiti, et ei oska öelda (hinne “4”);
- 10 korral, leiti et on plagiaat (hinne “6” või “7”).

Seega olid mentorid küllaltki veendunud, et tegemist on plagiaadiga. Otsust kinnitas peamiselt see, et mõlema programmi vastavate ridade töö on ühesugune, tehtud oli vaid kosmeetilisi muudatusi, millest kõige enam paistis silma muutujate nimede vahetamine. Siit võib järeldada, et üldiselt on väga ühesuguse struktuuriga ülesande puhul mentorite otsus kattuv, ometigi leidus üks, kes siiski ei suutnud kindlat otsust selle paari puhul teha.

4.5 Identsete muutujanimede ja erineva struktuuriga paar

o9.py

```
1  nimi = str(input("Sisestage failinimi: "))
2
3  fail = open(nimi, encoding="UTF-8")
4
5  laulud = []
6
7  for rida in fail:
8      if "\n" in rida:
9          laulud.append(str(rida[0:-1]))
10     else:
11         laulud.append(str(rida))
12
13 fail.close()
14
15 print("Muusikapalade valik: ")
16 a = 0
17 for i in laulud:
18     print(str(a+1)+". "+laulud[a])
19     a += 1
20 b = int(input("Valige laulu järjekorranumber: "))
21 print("Mängitav muusikapala on " + laulud[b-1] + ".")
```

o25.py

```
1  nimi = input("Palun sisestage failinimi: ")
2  fail = open(a)
3  laulud = []
4  i = 0
5
6  print("Muusikapalade valik: ")
7
8  for rida in fail:
9      i += 1
10     print(str(i) + ". " + rida)
11     b += [rida]
12
13 fail.close()
14
15 a = int(input("Valige laulu järjekorranumber: "))
16 b -= 1
17 print("Mängitav muusikapala on " + laulud[b])
```

Joonis 8. Lahendused o9 ja o25.

Siinkohal on tegemist ühe reaalselt kodutööna esitatud lahenduse (o9) ja ühe modifitseeritud lahendusega (o25). Paari on kujutatud joonisel 8. Kuna ühesuguse struktuuri, kuid erinevate muutujanimedega lahenduste puhul kahtlustatakse pigem plagiaati, sooviti uurida, kuidas hinnatakse lahendusi, mille puhul on olukord vastupidine - lahenduste struktuur on erinev, kuid nimed on identsed. Seetõttu võeti üks esitatud lahendus ning muudeti ära selle muutujanimed nõnda, et need langeksid kokku teises lahenduses (o9) kasutatutega. Selle paari puhul oodati, et mentoreid võib segadusse ajada muutujanimede identne kattuvus, kuid kuna tegemist on loogiliste nimedega (mis väljendavad muutuja sisu) ning tehniline lahendus on erinev, püstitati hüpotees, et lahendusi pigem ei peeta plagiaadiks. Järgnevalt on toodud mentorite hinnangud käesolevale paarile.

4.5.1 Argumendid plagiaadi poolt

Küsitluses paluti selgitada, milliseid argumente saab esitada selle väite poolt, et tegemist on plagiaadiga. Siinkohal toodi välja järgnevaid aspekte.

- Sarnasus faili avamisel.
- Üks tsükkel on sarnane (vasakpoolsel lahendusel read 17–19, parempoolsel lahendusel read 8–11).
- Sama kasutajale kuvatav tekst.
- Ühesugused muutujad “laulud”, “nimi”, “fail”, “a”, “b”.
- Ühesugused for-tsükli esimesed read.
- Sarnasus avaldub ridades 1 ja 1 (failinime küsimine); 5 ja 3 (“laulud” nimelise järjendi loomine); 20 ja 15 (kasutajalt laulu numbri küsimine); 21 ja 17 (tulemuse väljastamine).

Kolmel korral toodi välja ka vaid, et seda ei peeta plagiaadiks. Üldine arvamus oli, et veidi sarnasusi on, kuid need tulenevad pigem ülesande lihtsusest. Üks leidis, et õppija on võib-olla küsinud nõu, kuid plagiaadiga tegemist kindlasti pole.

4.5.2 Argumendid plagiaadi vastu

Küsitluses paluti selgitada, milliseid argumente saab esitada selle väite poolt, et tegemist ei ole plagiaadiga. Siinkohal toodi välja järgnevaid aspekte.

- Parempoolsel lahendusel on ridadel 15–17 viga, kuna ei kasutata järjekorranumbrit, vasakpoolsel lahendusel on aga õigesti (ridadel 20–21).
- Tsükli sisu on oluliselt erinev.
- Erinev koodiridade arv.
- Ühes lahenduses on üks tsükkel, teises kaks tsüklit.
- Ühel juhul on `input()` veel eraldi sõneks teisendatud, teise puhul mitte.
- Järjendisse lisamine on erinev.
- Failist andmete lugemine on lahendatud erinevalt.
- Kasutajat failinime küsimiseks väljastatav tekst on erinev.

Lisaks kommenteeriti, et asjaolu, et for-tsüklite struktuur on erinev, on oluline, kuna selleks, et osata seda erinevalt esitada, peab sisust siiski aru saama (seega poleks plagiaadiks põhjust).

4.5.3 Võimalikud viisid jälgede segamiseks

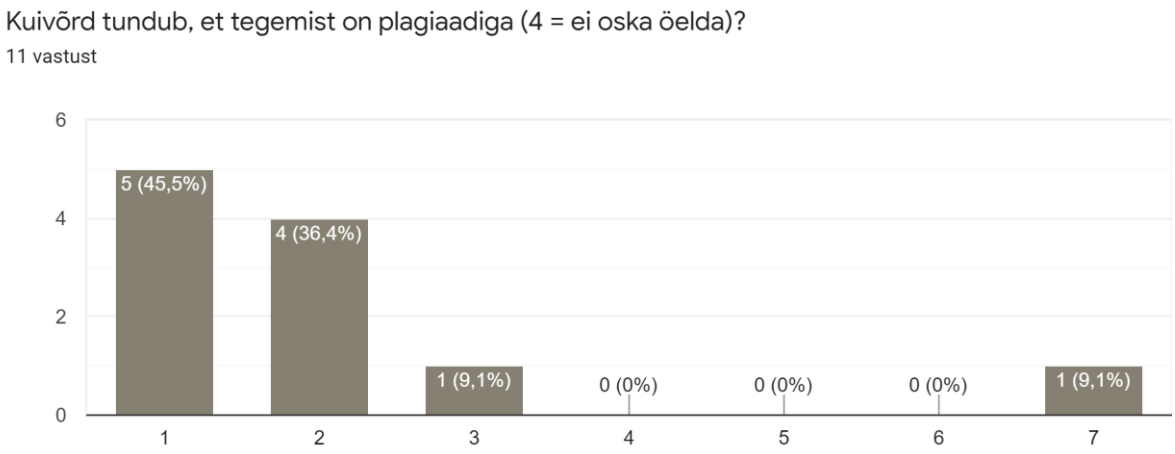
Küsitluses paluti selgitada, milliseid tehnikaid on võidud kasutada jälgede segamiseks. Siinkohal toodi välja järgnevaid aspekte:

- ühe tsükli lisamine;
- muutujad “a” ja “b” omavahel vahetatud;
- lisatud tühjad read;
- kood on läbi töötatud ja selle struktuur teisendatud;
- muutuja “b” väärtust on muudetud enne print-käsku.

Kolmel juhul kommenteeriti vaid, et ei peeta plagiaadiks. Lisati ka, et kui tegemist peaks olema plagiaadiga, siis on kopeerija teinud nõnda palju muudatusi, et see sisuliselt vabastaks ta süüst.

4.5.4 Hinnang sellele, kuivõrd on tegemist plagiaadiga

Mentoritel paluti hinnata seitsmepalliskaalal (kus 1 = kindlasti pole plagiaat, 4 = ei oska öelda, 7 = kindlasti on plagiaat), kuivõrd tundub, et tegemist on plagiaadiga.



Joonis 9. Mentorite hinnang plagiaadi osas, 1 = kindlasti pole plagiaat ja 7 = kindlasti on plagiaat.

Saadud vastuste jaotust illustreerib joonis 9. Kokkuvõtvalt võib öelda, et:

- 10 juhul leiti, et ei ole plagiaat (hinne “1”,“2” või “3”);
- 1 juhul leiti, et on plagiaat (hinne “6” või “7”).

Seega oli suurem osa mentoritest ühel meelel, et tegemist pole plagiaadiga. Peamisteks põhjusteks peeti seda, et lahenduse sisu on võrdlemisi erinev, tsüklite arv on erinev ning parempoolne lahendus on vigane, samas kui vasakpoolne pole. Leidus siiski üks mentor, kes oli veendunud, et tegemist on plagiaadiga. Tema peamiseks argumendiks plagiaadi poolt oli, et mõlemas lahenduses kasutatakse järjendit “laulud”, mida kasutatakse ka vastuse välja trükkimisel, mainiti ka sarnasust faili avamisel ja ekraanile trükkimisel, kuid nende puhul leiti, et sarnasused tulenevad ülesandest. Plagiaadi vastu rääkiva argumendina toodi välja, et muutuja “nimi” juures on püütud kasutada teist teksti ning arvati, et jälgede segamiseks on teises lahenduses proovitud jätta teine tsükkel ära ning muutuja “b” väärtust on muudetud teises kohas. Üldiselt langesid mentorite hinnangud siiski kokku ootustega, sarnased muutujanimed üksi pole piisav põhjus plagiaadi kahtlustamiseks, eriti kui programmide struktuur on erinev.

4.6 Osalt erinevate muutujanimede ja osalt identsete ridadega paar

o22.py

```
failinimi = input("Palun sisestage failinimi: ")

fail = open(failinimi, encoding="UTF-8")
laulud = []

for rida in fail:
    laul = rida
    if laul[-1:] == "\n":
        laul = laul[:-1]
    laulud.append(laul)
fail.close()

print("Muusikapalade valik:")
for i in range (0, len(laulud)):
    print(str(i + 1) + ".", laulud[i])
number = input("\nValige laulu järjekorranumber: ")

pala = laulud[int(number) - 1]
print("Mängitav muusikapala on", pala + ".")

```

o7.py

```
muutuja = input("Palun sisestage failinimi: ")
fail = open(muutuja, encoding = "UTF-8")
laulud = []
for rida in fail:
    laul = rida
    if laul[-1:] == "\n":
        laul = laul[:-1]
    laulud.append(laul)

print("Muusikapalade valik:")
for i in range(len(laulud)):
    print(str(i + 1) + ".", laulud[i])

muutuja2 = int(input("Valige laulu järjekorranumber: ")) - 1
print("Mängitav muusikapala on " + str(laulud[muutuja2]))

fail.close()

```

Joonis 10. Lahendused o22 ja o7.

Lahenduste o22 ja o7 (joonisel 10) analüüsimisel tundus autorile, et tegemist ei ole selgelt identsete lahendustega, kuid leidus küllaltki suuri sarnasusi. Mõlemad tsüklid on üsna identsed, teisel juhul on vasakpoolsel lahendusel lisatud vaid ebavajalik nullist alustamine (real 14). Read 1–4 vasakpoolsel lahendusel ja read 1–3 parempoolsel lahendusel on samuti sarnased, erinevus on vaid ühes muutujanimes, kuid see võib olla ka juhuslik. Peamine erinevus näib olevat kasutajalt laulu järjekorranumbri küsimises ja mängitava muusikapala ekraanile trükkimises. Nendel põhjustel leidis autor, et on võimalik, et lahendusi on omavahel detailideni arutatud või teatud käske kopeeritud. Järgnevalt on toodud mentorite hinnangud käesolevale paarile.

4.6.1 Argumendid plagiaadi poolt

Küsitluses paluti selgitada, milliseid argumente saab esitada selle väite poolt, et tegemist on plagiaadiga. Siinkohal toodi välja järgnevaid aspekte.

- Lahenduse algus on väga sarnane (kuid samas lõpp on veidi unikaalsem).
- Vasakpoolse lahenduse read 1–15 on identsed parempoolse lahenduse ridadega 1–12.
- Esimene for-tsükkel on mõlemal lahendusel ühesugune.
- Teine for-tsükkel on ühesugune, ainus erinevus on üleliigne argument 0.
- Mõlemal juhul on failist lugemisel kasutatud `if laul[-1] == „\n“` (ridadel 8 ja 6), mis pole väga levinud lahendus.
- Muudetud on vaid muutujate nimesid.
- Ekraanile kuvatav tekst on ühesugune.
- Real 13 vasakpoolsel lahendusel ja real 10 parempoolsel lahendusel on samasugune lohakusviga: kui muidu on väljastatava rea lõppu (enne kasutajalt saadavat sisendit) jäetud tühik, siis siin pole kummalgi juhul seda tehtud.

Kommenteeriti ka, et muutujate nimed on küll erinevad, kuid kirjutatud kood on liiga sarnane. Peamiselt veenis see, et kõige keerukamad kohad on lahendustes kas identsed või väga väikeste erinevustega.

4.6.2 Argumendid plagiaadi vastu

Küsitluses paluti selgitada, milliseid argumente saab esitada selle väite poolt, et tegemist ei ole plagiaadiga. Siinkohal toodi välja järgnevaid aspekte.

- Fail suletakse erinevas kohas.
- Muutujate nimed on erinevad.
- Kasutaja poolt muusikapala valimine ja selle ekraanile kuvamine on lahendatud veidi erinevalt.
- Int-käsklust on kasutatud erinevalt (vasakpoolsel lahendusel real 16, parempoolsel lahendusel real 14).

Viiel korral kommenteeriti vaid, et pole argumente, kuna peetakse plagiaadiks. Ühel korral kommenteeriti ka, et pigem võiks olla tegemist plagiaadiga parempoolse lahenduse puhul, kuna vasakpoolne tundub loogilisema ülesehitusega.

4.6.3 Võimalikud viisid jälgede segamiseks

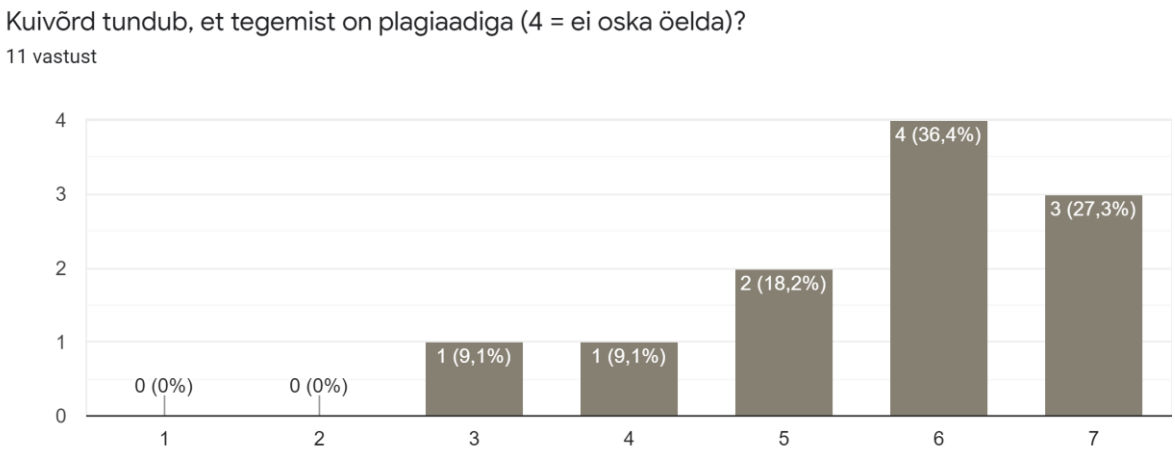
Küsitluses paluti selgitada, milliseid tehnikaid on võidud kasutada jälgede segamiseks. Siinkohal toodi välja järgnevaid aspekte:

- muutujate nimesid muudetud;
- lõppu veidi optimeeritud;
- faili sulgemise lauset liigutatud;
- tühjade ridade lisamine/eemaldamine.

Üheks võimaluseks peeti, failist lugemine on tehtud koos, kuid lihtsamad osad on kirjutatud eraldi.

4.6.4 Hinnang sellele, kuivõrd on tegemist plagiaadiga

Mentoritel paluti hinnata seitsmepalliskaalal (kus 1 = kindlasti pole plagiaat, 4 = ei oska öelda, 7 = kindlasti on plagiaat), kuivõrd tundub, et tegemist on plagiaadiga.



Joonis 11. Mentorite hinnang plagiaadi osas, 1 = kindlasti pole plagiaat ja 7 = kindlasti on plagiaat.

Saadud vastuste jaotust illustreerib joonis 11. Kokkuvõtvalt võib öelda, et:

- 1 juhul leiti, et pole plagiaat (hinne “3”);
- 1 juhul ei osatud öelda (hinne “4”);
- 8 juhul leiti, et on plagiaat (hinne “5”, “6” või “7”).

Seega oli enamuse hinnang, et käesoleva paari puhul võiks pigem olla tegemist plagiaadiga. Üks mentor kaldus veidi enam selle poole, et pole tegemist plagiaadiga, tema hinnangul oli tegemist päris keerulise juhtumiga, kuna lahenduse algus on väga sarnane, kuid lõpp on unikaalsem. Mentor, kes andis plagiaadile hindeks “4” ehk “ei oska öelda”, ei lisanud ka argumente plagiaadi poolt ega vastu, puudusid ka viisid jälgede segamiseks. Üldiselt veenis mentoreid kõige enam asjaolu, et just keerulisemad kohad olid väga sarnaselt lahendatud. Seega langes üldjoontes mentorite hinnang kokku autori omaga.

4.7 Sarnaste muutujanimede ja identse struktuuriga paar

o9.py

```
nimi = str(input("Sisestage failinimi: "))

fail = open(nimi, encoding="UTF-8")

laulud = []

for rida in fail:
    if "\n" in rida:
        laulud.append(str(rida[0:-1]))
    else:
        laulud.append(str(rida))

fail.close()

print("Muusikapalade valik: ")
a = 0
for i in laulud:
    print(str(a+1)+". "+laulud[a])
    a += 1
b = int(input("Valige laulu järjekorranumber: "))
print("Mängitav muusikapala on " + laulud[b-1] + ".")
```

o19.py

```
fnimi = str(input("Palun sisestage failinimi: "))
fail = open(fnimi, encoding="UTF-8")
laulud = []
for rida in fail:
    if "\n" in rida:
        laulud.append(str(rida[0:-1]))
    else:
        laulud.append(str(rida))
fail.close()
print("Muusikapalade valik:")
a = 0
for i in laulud:
    print(str(a+1)+". "+laulud[a])
    a += 1
num = int(input("Valige laulu järjekorranumber: "))
print("Mängitav muusikapala on "+laulud[num-1]+".")

```

Joonis 12. Lahendused o9 ja o19.

Lahenduste o9 ja o19 (joonisel 12) puhul on tegemist selgelt identse struktuuriga lahendustega. Kui eemaldada vasakpoolsest lahendusest tühjad read, oleks iga rea funktsioon ühesugune. Erinevused seisnevad vaid mõne muutuja nimes, lisaks on kasutajale kuvatud veidi erinev tekst faili nime küsimisel. Lisaks sellele on mõlemal juhul kasutatud esimesel real üleliigset str-funktsiooni. Nendel põhjustel on autori hinnang, et tegemist on kindlasti plagiaadiga. Järgnevalt on toodud mentorite hinnangud käesolevale paarile.

4.7.1 Argumendid plagiaadi poolt

Küsitluses paluti selgitada, milliseid argumente saab esitada selle väite poolt, et tegemist on plagiaadiga. Siinkohal toodi välja järgnevaid aspekte:

- programmid on pea identsed;
- iga samm on ühesugune;
- sisuliselt on lahendused ühesugused, erinevus on vaid kahe muutuja nimes ja tühjades ridades.

Siinkohal olid mentorid selgelt üksmeelel, kommenteeriti, et tegemist on üks-ühele lahendusega ning et on väga väike võimalus, et selline asi juhtub juhuslikult .

4.7.2 Argumendid plagiaadi vastu

Küsitluses paluti selgitada, milliseid argumente saab esitada selle väite poolt, et tegemist ei ole plagiaadiga. Siinkohal toodi välja järgnevaid aspekte:

- kahe muutuja nimed on erinevad;

- ühes lahenduses on struktureerimiseks kasutatud tühje ridasid.

Seitse mentorit kommenteerisid vaid, et sellised argumendid puuduvad. Lisati ka, et erinevused failide vahel on minimaalsed ja tehtud vaid selleks, et tekitada mingeid erinevusi Ühel korral pakuti siiski välja ka võimalus, et on õpitud sarnaste näidisprogrammide järgi.

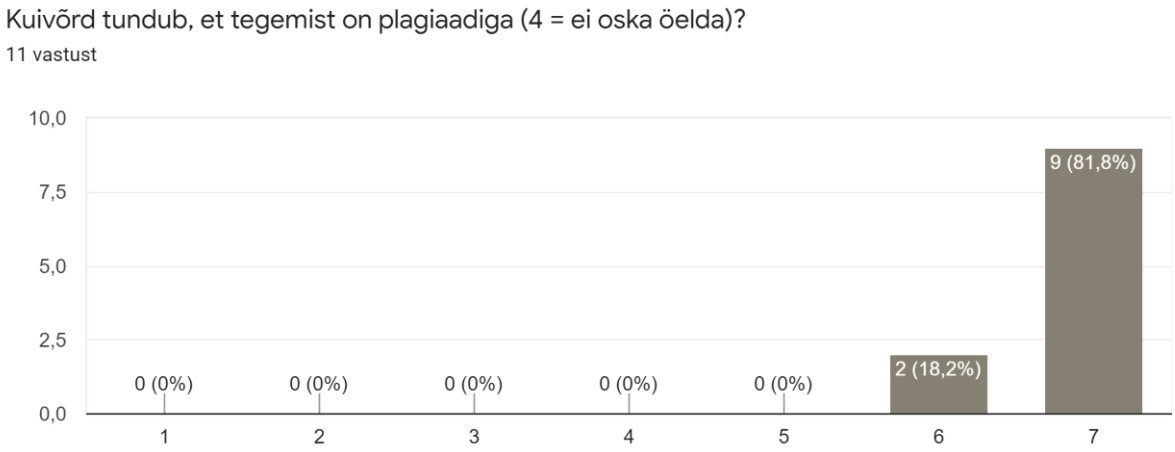
4.7.3 Võimalikud viisid jälgede segamiseks

Küsitluses paluti selgitada, milliseid tehnikaid on võidud kasutada jälgede segamiseks. Siinkohal toodi välja järgnevaid aspekte:

- muutujate nimede vahetamine;
- muudeti kasutajale kuvatava küsimuse teksti;
- eemaldatud/lisatud tühjad read, et luua visuaalselt mulje teistsugusest programmist.

4.7.4 Hinnang sellele, kuivõrd on tegemist plagiaadiga

Mentoritel paluti hinnata seitsmepalliskaalal (kus 1 = kindlasti pole plagiaat, 4 = ei oska öelda, 7 = kindlasti on plagiaat), kuivõrd tundub, et tegemist on plagiaadiga.



Joonis 13. Mentorite hinnang plagiaadi osas, 1 = kindlasti pole plagiaat ja 7 = kindlasti on plagiaat.

Saadud vastuste jaotust illustreerib joonis 13. Kokkuvõtvalt võib öelda, et:

- 11 korral ehk kõigi mentorite hinnangul on tegemist plagiaadiga (hinne “6” või “7”).

Selle paari puhul olid mentorite kommentaarid ja hinnangud väga ühtsed. Leiti, et kuna kogu sisuline osa koodist on identne, erinevus on vaid paari muutuja nimes ja tühjades ridades, on tegemist kindlasti plagiaadiga. Seega langes hinnang kokku autori omaga.

4.8 Erinevate muutujanimede ja struktuuriga paar

o13.py

```
1  a=str(input())
2  file=open(a)
3  m=1
4  musa=[]
5  print("Muusikapalade valik:")
6  for i in (file):
7      musa.append(i)
8      print(str(m)+".",i)
9      m=m+1
10 b=int(input("Valige laulu järjekorranumber: "))
11 b=b-1
12 print("Mängitav muusikapala on "+ musa[b])
```

o21.py

```
1  failinimi = input("Palun sisestage failinimi: ")
2
3  fail = open(failinimi, encoding = "UTF-8")
4  lood = []
5  for rida in fail:
6      lood.append(str(rida))
7  fail.close()
8
9  print("Muusikapalade valik:")
10 print()
11 for number, täht in enumerate(lood, 1):
12     print(str(number) + ".", täht)
13 print()
14 nr = int(input("Valige laulu järjekorranumber: "))
15 lugu = -1 + nr
16 print("Mängitav muusikapala on " + lood[lugu])
```

Joonis 14. Lahendused o13 ja o21.

Lahenduste o13 ja o21 (joonisel 14) puhul on tegemist küllaltki erinevate lahendustega, vasakpoolne fail paistab küllaltki minimalistlik, samas kui parempoolne fail on mitme rea võrra pikem. Erinev on ka tsüklite arv - vasakpoolsel on üks for-tsükkel, parempoolsel lahendusel aga kaks. Lahenduste vahel leidub küll sarnasusi, kuid tundub, et need tulenevad pigem lihtsast ülesandest ja näidisväljundist. Parempoolsel lahendusel on kasutatud teises for-tsüklis enumerate-funktsiooni, mis näitab kas suuremat oskust või huvi. Nendel põhjustel on autori hinnang, et tegemist pole plagiaadiga. Järgnevalt on toodud mentorite hinnangud käesolevale paarile.

4.8.1 Argumendid plagiaadi poolt

Küsitluses paluti selgitada, milliseid argumente saab esitada selle väite poolt, et tegemist on plagiaadiga. Siinkohal toodi välja järgnevaid aspekte.

- Vasakpoolse lahenduse read 10–12 on sarnased parempoolse lahenduse ridadega 14–16.
- Vasakpoolse lahenduse read 4, 6 ja 7 on sarnased parempoolse lahenduse ridadega 4, 5 ja 6 (failist lugemine ja järjendisse lisamine).
- Parempoolse lahenduse puhul on kasutatud teist for-tsüklit (ridadel 11–12), mille vajalikkus pole selge, võib-olla on tahetud muuta kood erinevaks.

Seitsmel korral kommenteeriti vaid, et pole argumente, kuna tegemist pole plagiaadiga. Lisati ka, et kasutajale kuvatav tekst on sama, kuid see tuleneb programmi näitest.

4.8.2 Argumendid plagiaadi vastu

Küsitluses paluti selgitada, milliseid argumente saab esitada selle väite poolt, et tegemist ei ole plagiaadiga. Siinkohal toodi välja järgnevaid aspekte.

- Tsüklites on erinevad tegevused, kusjuures parempoolsel lahendusel on ka teine tsükkel, kus kasutatakse keerulisemaid funktsioone (enumerate-funktsioon).
- Muutujate nimed on erinevad.
- Ridade arv on erinev.
- Faili avamine on tehtud erinevalt.
- Muusikapalade valik on tehtud erinevalt.
- Käekiri ehk kasutatav stiil on erinev.

Kommenteeriti ka, et selliste erinevuste tekitamine nõuab juba omajagu oskusi, et programm ikkagi sama asja teeks, mis poleks aga algajale jõukohane. Lisaks on raske leida põhjust, miks peaks keegi, kes seda oskab, tegelema nii lihtsa programmi maha kirjutamisega. Lisati ka, et kummagi lahenduse puhul on lähenemine erinev: üks on kirjutatud programmeerija poolt, teine matemaatiku poolt, kuna ühel juhul on vaja kõik lahti kirjutada.

4.8.3 Võimalikud viisid jälgede segamiseks

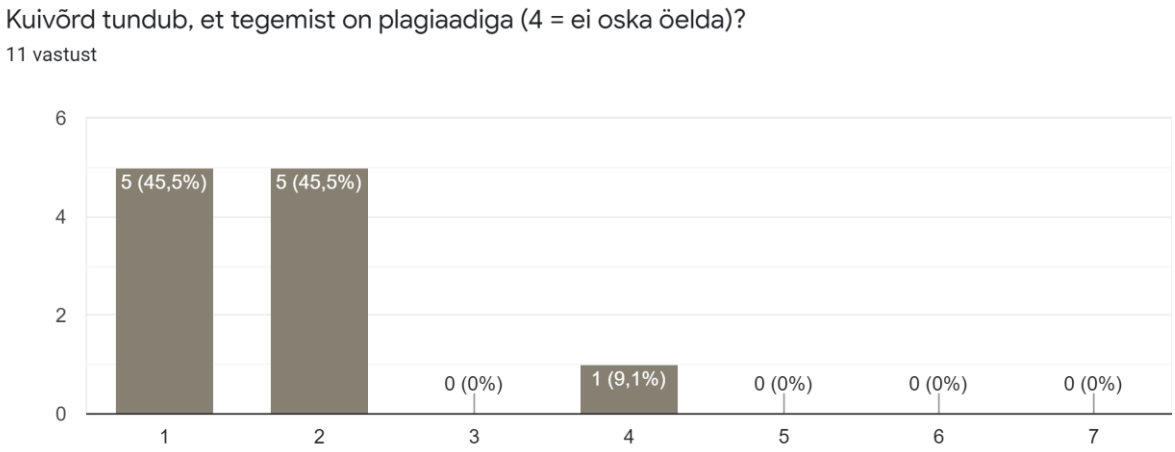
Küsitluses paluti selgitada, milliseid tehnikaid on võidud kasutada jälgede segamiseks. Siinkohal toodi välja järgnevaid aspekte:

- vahetatud muutujate nimesid;
- parempoolse lahenduse real 15 on raske uskuda, et keegi kirjutab "`-1+ nr`", see võib olla kunstlik meelega tehtud muudatus visuaalsete erinevuste tekitamiseks.

Rohkem kommenteeriti siiski, et peetakse ebatõenäoliseks, et keegi näeks nõnda palju vaeva lahenduse ümber kirjutamiseks, kuuel korral kommenteeriti jällegi vaid, et seda ei peeta plagiaadiks.

4.8.4 Hinnang sellele, kuivõrd on tegemist plagiaadiga

Mentoritel paluti hinnata seitsmepalliskaalal (kus 1 = kindlasti pole plagiaat, 4 = ei oska öelda, 7 = kindlasti on plagiaat), kuivõrd tundub, et tegemist on plagiaadiga.



Joonis 15. Mentorite hinnang plagiaadi osas, 1 = kindlasti pole plagiaat ja 7 = kindlasti on plagiaat.

Saadud vastuste jaotust illustreerib joonis 15. Kokkuvõtvalt võib öelda, et:

- 10 korral leiti, et pole plagiaat (hinne “1” või “2”);
- 1 korral leiti, et ei oska öelda (hinne “4”).

Seega oli enamuse selge arvamus, et selle paari puhul pole tegemist plagiaadiga. Mentor, kes vastas, et ei oska öelda, pidas kummaliseks rida 15 (kus oli kasutatud “`-1 + nr`”, kuigi tavaliselt kasutatakse “`nr - 1`”) ning asjaolu, et parempoolsel lahendusel on kasutatud teist for-tsüklit, mille vajalikkus jääb ebaselgeks. Üldiselt peeti kõige olulisemaks seda, et lisaks sellele, et lahendustes on muutujate nimed erinevad, on ka kood sisuliselt niivõrd erinev, et selle ümber kirjutamine nõuaks palju vaeva ja oskusi. Seega langes mentorite hinnang kokku autori omaga.

4.9 Üldised argumendid plagiaadi poolt ja vastu

Kõikide paaride peale kokku saab tuua välja peamised tegurid, mida lahenduste sarnasuse hindamisel vaadatakse. Järgnevalt selgitatakse, kuidas need üldisi hinnanguid mõjutavad ning kuidas neid saab kasutada kas argumentideks plagiaadi poolt või vastu.

4.9.1 Muutujate nimed

Muutujate nimed on aspekt, mida alati välja tuuakse, kuna seda saab iga koodi puhul vaadata. Erinevusi muutujate nimedes saab tuua üheks argumendiks plagiaadi vastu. Ometigi kuna

tegemist on väga lihtsalt teostatava muudatusega, siis üksinda pole see väga tugev argument, vaid pigem aitab hinnangut kinnitada, kui erinevusi seisneb ka teistes keerukamates aspektides. Kui mujal erinevusi pole, siis näitavad erinevad muutujate nimed pigem seda, et on püütud lihtsate meetmetega jälgi segada. Ka küsitluses kasutatud paaride hulgas leidus üks, mille puhul pea ainsaks erinevuseks olid muutujate nimed ning kõik mentorid olid üksmeelel, et sellisel juhul võiks tegemist olla plagiaadiga.

Lisaks vaadatakse muutujate nimede puhul ka seda, kuivõrd need kajastavad muutujate sisu. Loomulikult ei tähenda see seda, et näiteks ühetähelised muutujate nimed peaksid kohe kahtlust äratama. Kuid kui on kaks lahendust, mis paistavad sarnased ning mõlemal juhul on kasutatud nt ebaloogilisi ühetähelisi muutujaid, siis see võib veelgi suurendada kahtlust, et read on teise tööst kopeeritud (ning sisust pole ise aru saadud), või vähemalt pole nähtud palju vaeva jälgede peitmisega või oma panuse lisamisega.

4.9.2 Lahenduste pikkus

Järgmine aspekt, mida lahenduste puhul hinnata, on nende pikkus. Ühesugune ridade arv tekitas mentorites kahtlust, et on koodi on kas kopeeritud või on ülesannet koos lahendatud. Üks viis, mida siinjuures jälgede peitmiseks kasutatakse, on tühjade ridade lisamine või nende äravõtmine. Tühjade ridade olemasolu teeb koodi mõistmise kergemaks, kuna struktuuri osad eristuvad kergemini, kuid vaid selle lisandiga mentoreid petta pole siiski võimalik. Isegi kui see mõjutab lahenduste pikkust või esmapilgul veidi ka üldist välimust, siis pettust sellega peita pole võimalik.

Teine viis jälgede peitmiseks ning lahenduse pikkusega manipuleerimiseks on kopeeritud koodi lühendamine või sinna ebavajalike osade lisamine. Koodi pikendamine asjassepuutumatute ridade või tsüklite lisamisega on tehnika poolest veidi kergem, kuna siin saab kasutada näidistes või varasemates lahendustes olnud koodi. Seetõttu on see veidi tõenäolisem, kuid mentorid leidsid, et see on küllaltki ebatavaline juhus. Küsitluses pakuti kahe paari (osalt erinevate muutujanimede ja osalt identsete ridadega paari ja sarnaste muutujanimede kuid erineva struktuuriga paari) puhul, et on võimalik, et plagiaadi varjamiseks on lahendusi optimeeritud. Samas jäädi pigem seisukohale, et koodi optimeerimine vajab juba oskusi ja teadmisi - kui õppijal on need olemas, siis poleks ka põhjust kellegi teise käest koodi kopeerida (vähemalt selliste lihtsamate ja lühemate ülesannete puhul). See võib olla viis, mida kasutatakse pikemate ülesannete puhul, kui näiteks tervet ülesannet ise ei lahendata, kuid osatakse spetsiifiliste endale tuttavate funktsioonide puhul kasutada lühemaid esitusi.

4.9.3 Programmide struktuur

Kolmas aspekt, mida sarnasuse hindamisel jälgiti, oli programmide struktuur. Selgeks plagiaadi märgiks peeti seda, kui programmi töö käik oli vastavatel ridadel ühesugune, s.t kui näiteks tühjade ridade eemaldamisel ja reakaupa lahenduste vaatamisel selgub, et igal real tehti ühesugune tegevus. See on aspekt, mis paistab silma ja tekitab kahtlust ka siis, kui on tehtud väiksemaid muudatusi jälgede peitmiseks, nt tühjade ridade või tühikute lisamine, muutuja väärtuse ühe võrra suurendamise/vähendamise tehnika muutmine, muutujanimede vahetamine, üleliigsete sulgude lisamine. Veidi suurem võimalik muudatus on siinkohal ka ridade asukoha muutmine seal, kus võimalik. Seda saab teha näiteks faili sulgemise puhul, kuna programmi tööd see ei mõjuta (eeldusel, et fail suletakse siiski peale sealt sisu lugemist). Vaid selliste muudatuste olemasolu ei saa jällegi plagiaadi välistada.

Programmi struktuuri muutmiseks on võimalik mängida ka muutujate ja tsüklite arvuga. Muutujate lisamine on selgelt lihtsam viis, kuid suurem muutujate arv võib olla põhjustatud ka sellest, et neid kasutati lahendamise käigus, kuid ideede muutumise tulemusena neid lõplikus lahenduses vaja ei läinud ja üleliigsed osad unustati enne esitamist ära kustutada. Tsüklite lisamine on veidi keerulisem. Siinkohal on võimalik lisada üleliigne tsükkel (nagu mainitud eelmises peatükis), või jaotada ühes tsüklis tehtav töö kahe tsükli vahel. Viimane ei vaja ka liiga palju oskusi, kuna seda on võimalik teha vaid esialgse tsükli uuesti kopeerimisega ja seal tehtavate tegevuste ära jaotamisega. Samas peeti seda pigem vähetõenäoliseks, kuna see ei pruugi siiski olla algajale jõukohane ning ei nähta põhjust, miks peaks keegi, kes seda oskab, tegelema nii lihtsa programmi mahakirjutamisega.

4.9.4 Programmide sisu

Programmide sarnasuse hindamisel on oluline vaadata lahenduste sisu. Kõige olulisemaks pidasid mentorid seda, kui erinevused seisnesid keerukamates osades, kuna nende osas on tõenäolisem, et on otsustatud kellegi teise lahendust kasutada. Sellise algajate kursuse ülesande puhul on keerukamateks kohtadeks failist lugemine, tsüklid ja järjenditega toimetamine. Siinkohal on oluline arvestada siiski ka sellega, et suure tõenäosusega kasutatakse abiks õpiku näiteid või varem lahendatud ülesandeid. Seetõttu ei tähenda suur sarnasus siin tingimata seda, et on kasutud lubamatuid võtteid, eriti situatsioonis, kus tegemist on standardlahendusega. Kahtlust äratab aga kindlasti see, kui lahendustes esinevad identsed vead.

Teine aspekt, mida sisu hindamisel jälgida, on autori stiil, mis tavaliselt lahendusesiseselt drastiliselt ei muutu. On võimalik võrrelda lahendust ka sama autori eelnevalt esitatud töödega, et näha, kas autori käekiri on ühtäkki muutunud. Siinkohal saab jälgida tühikute, tühjade ridade, täpitähtede, suure algustähe kasutust, muutuja väärtuse ühe võrra muutmise tehnikat ja seda, kas teksti kujutamiseks kasutatakse ülakomasid või jutumärke. Algajatele mõeldud kursuse korral tuleb siiski arvestada sellega, et isikupärane stiil pole tingimata veel kujunenud ning kui eeskuju võetakse erinevas stiilis näidetest, siis võib väikseid erinevusi ka lahenduses siiski esineda.

Mitmel korral toodi välja ka seda, et jäi mulje, justkui oleks hakatud kopeerima, kuid sellest oli ühel hetkel siiski loobutud - sellist käitumist plagiaadiks ei peetud. Juhul, kui koodi oli esialgu kopeeritud, kuid seejärel struktuuris suuri muudatusi tehtud ja sellega väga palju vaeva nähtud, arvati samuti, et tegemist poleks enam plagiaadiga.

4.10 Üksmeel hinnangutes

Mentorite hinnangute osas leidus paare, mille puhul olid kõik veendunud, et tegemist on/pole plagiaadiga, kuid esines ka erimeelsusi. Hinnangute jaotus kõikide paaride osas on toodud tabelis 1.

Kuuest paarist viiel korral oli mentorite arvamus suuremas osas ühesugune, kuid nendest kolmel juhul leidus siiski ka üks mentor, kes ei osanud hinnangut anda. Kahel juhul leidus üks mentor, kelle arvamus oli vastupidine, seda sarnaste muutujanimede kuid erineva struktuuriga paari ning identsete muutujanimede ja erineva struktuuriga paari puhul. Mõlemal juhul oli enamuse arvamus, et tegemist pole plagiaadiga ning mõlemal juhul oli sama mentor tugeval arvamusel, et tegemist on plagiaadiga. Arvamused lahknesid veidi ka osalt erinevate muutujanimede ja osalt identsete ridadega paari puhul, kus enamus kaldus siiski pigem plagiaadi poole, kuid suured erinevused olid kindluse astmes (hindeks kas “5”, “6” või “7”), lisaks oli üks mentor, kes kaldus veidi selle poole, et tegemist pole plagiaadiga.

Tabel 1. Mentorite hinnangud plagiaadile kuue lahendustepaari osas.

Hinnang **Paar**	**1 - kindlasti pole plagiaat**	**2**	**3**	**4 - ei oska öelda**	**5**	**6**	**7 - kindlasti on plagiaat**
Sarnaste muutujanimede, kuid erineva struktuuriga paar	4 (36,4%)	5 (45,5%)		1 (9,1%)		1 (9,1%)	
Sarnaste muutujanimede ja struktuuriga paar				1 (9,1%)		4 (36,4%)	6 (54,5%)
Identsete muutujanimede ja erineva struktuuriga paar	5 (45,5%)	4 (36,4%)	1 (9,1%)				1 (9,1%)
Osalt erinevate muutujanimede ja osalt identsete ridadega paar			1 (9,1%)	1 (9,1%)	2 (18,2%)	4 (36,4%)	3 (27,3%)
Sarnaste muutujanimede ja identse struktuuriga paar						2 (18,2%)	9 (81,8%)
Erinevate muutujanimede ja struktuuriga paar	5 (45,5%)	5 (45,5%)		1 (9,1%)			

Üldiselt saab öelda, et kui tegemist on lahendustega, kus muudatused seisnevad vaid muutujate nimedes ja tühjades ridades või tühikutes, on mentorid selgel üksmeelel, et tegemist on plagiaadiga. Keerulisemate juhtumite puhul on aga nende hulgas erimeelsusi. Erinevused kindluse astmes võivad olla põhjustatud ka madalamast enesekindlusest plagiaadikahtlustuse esitamise osas. Siit võib järeldada, et mentorite ettevalmistust saaks programmide sarnasuse tunnetuse osas suurendada nt õppematerjalide ja näidete abiga.

4.11 Hindamise keerukus

Viimasena küsiti mentoritelt, kas küsimustele oli lihtne vastata või tekkis ka keerulisi olukordi või küsimusi ning milliste paaride/valikute juures. Kahe mentori hinnangul oli vastamine lihtne, kahe hinnangul keeruline. Spetsiifilistest paaridest toodi raskuse poolest kõige enam välja neljandat (osalt identsete muutujanimede ja osalt identsete ridadega paar, seda mainiti kolm korda) ja kuuendat (erinevate muutujanimede ja struktuuriga paar, seda mainiti kaks korda). Viimane on veidi üllatav, kuna selle paari puhul hindasid lõpuks kõik mentorid peale ühe, et pigem pole tegemist plagiaadiga, üks mentor ei suutnud otsustada. Kõige huvitavamateks paarideks pidas üks mentor esimest (sarnaste muutujanimede kuid erineva struktuuriga paar) ja viimast paari (erinevate muutujanimede ja struktuuriga paar).

Raskeks tegi hindamise see, et programmid võivad olla sarnased ka programmi töö näite tõttu, tõenäoliselt on tunnis lahendatud ka samu ülesandeid. Lisaks peeti üheks võimaluseks et kui peale vaadates võib tunduda, et rida on selgelt maha kirjutatud, kuna ei lähe kokku ülejäänud koodiga, on üks võimalus ka see, et algul läheneti ülesandele teisiti ning hiljem ununes mõni algse lahenduse rida esitatud koodi (arvati, et see võis juhtuda esimese paari puhul). Lisati ka, et kui pole tegemist äärmusega (nt täiesti üks-ühele lahendus), siis on hindamine keeruline. Kuna tegemist oli lihtsa ülesandega, on keeruline kirjutada koodi hästi omapäraselt, fantaasiat saab kasutada peamiselt muutujate nimede juures.

5. Kokkuvõte

Programmide sarnasuse määratlemine on oluline, kuid keerukas ülesanne. Bakalaureusetöö eesmärk oli tutvustada keerukuse põhjuseid ning näidata, kuidas sarnasust hinnatakse ja millistest aspektidest tulenevalt plagiaati kahtlustatakse. Selleks analüüsiti Tartu Ülikooli algajatele mõeldud veebipõhise kursuse "Tehnoloogia tarbijast loojaks" raames esitatud ühe ülesande 24 lahendust kolmes faasis. Esialgu vaadati programmid käsitsi läbi, seejärel kasutati Moodle'i automaatkontrolli vahendit vahendit ning kolmandaks moodustati saadud sarnaste lahenduste klastrite alusel kuus olemuselt erinevat lahenduste paari. Viimaseid paluti hinnata sama kursuse mentoritel nelja aspekti osas: milliseid argumente saaks tuua plagiaadi poolt, milliseid plagiaadi vastu, mida on võidud teha jälgede segamiseks ja milline on hinnang seitsmepalliskaalal sellele, kas tegemist võiks olla plagiaadiga.

Käsitsi lahenduste uurimise ja Moodle'i automaatse sarnasuse kontrolli tulemuste võrdlemisel selgus, et mõlemal juhul moodustusid sarnased klastrid. Käsitsi uurimisel tuli siiski välja ka paar, mis äratas kahtlust, kuid mille puhul olid sarnasused piisavalt peidetud, et automaatkontroll seda esile ei tõstnud. Kui mentoritel paluti sama paari hinnata, oli enamuse arvamus, et tegemist on plagiaadiga. Peamiseks kahtluse põhjuseks oli ebatavaline failist lugemise viis.

Küsitluse tulemuste põhjal võib öelda, et peamiselt kahtlustatakse sarnaste lahenduste puhul plagiaati siis, kui sarnasused esinevad keerukates kohtades. Lihtsalt muudetavateks aspektideks ja jälgede segamise viisideks peeti muutujate nimede vahetamist, tühjade ridade või tühikute lisamist, kasutajale kuvatava teksti muutmist. Struktuuris esinevad suuremad erinevused viitavad sellele, et ülesannet on lahendatud eraldi. Arvati ka, et kui lahendus on siiski kopeeritud, kuid lahenduse ülesehitust on oluliselt muudetud või lühendatud, vajab see vaeva ja oskusi, ning seetõttu pole enam tegemist plagiaadiga.

Üldiselt mentorite hinnangud plagiaadi osas kattusid, erinevusi oli rohkem kindluse astmes, mis võib viidata ka mentori enesekindlusele rangema hinnangu andmisel. Kahel juhul kuuest esines siiski ka olukord, kus üks mentor oli enamuse seisukohaga võrreldes tugeval vastupidisel arvamusel. Siit võib järeldada, et mentorite tunnetust sarnasuse hindamise osas saaks enne kursuse algust veidi enam õppematerjalide ja näidiste abiga ühtlustada.

Väiksem osa mentoritest tundis, et hindamine oli nende jaoks lihtne, suurema osa jaoks osutus see siiski keeruliseks. Probleemi üheks oluliseks allikaks on see, et kuna tegemist on algajatele

mõeldud kursusega, on ülesanne võrdlemisi lihtne ja lühike ning programmi kirjelduses on oodatavat lahendust detailselt kirjeldatud, seega loovusele väga palju ruumi ei jää, lisaks võivad lahendused sarnaneda ka seetõttu, et õpitakse ühesuguste näidete ja ülesannete põhjal.

Edaspidi saaks uurida seda, kuidas aitab või koormab mentoreid esitatud lahendusprotsesside täpsem analüüs, mida on võimalik teha nt programmeerimiskeskkonnas Thonny kogutud logide põhjal. Võib-olla oleks mõistlik luua selle jaoks vahend, mis automaatselt osa tööst ära teeks. Lisaks sellele oleks huvitav võrrelda kahtlust tekitavalt sarnaste programmide osakaalu veebipõhiste kursuste ja auditoorsel õppel põhinevate kursuste vahel ning vaadata, millised on Tartu Ülikooli üliõpilaste arusaamad plagiaadist, eriti programmeerimise puhul.

6. Viidatud kirjandus

Aasheim, C. L., Rutner, P. S., Li, L., & Williams, S. R. (2019). Plagiarism and programming: A survey of student attitudes. Journal of information systems education, 23(3), 5.

Cosma, G., & Joy, M. (2008). Towards a definition of source-code plagiarism. IEEE Transactions on Education, 51(2), 195–200.

Culwin, F., MacLeod, A., & Lancaster, T. (2001). Source code plagiarism in UK HE computing schools. Issues, Attitudes and Tools, South Bank University Technical Report SBU-CISM-01-02.

Dick, M., Sheard, J., Bareiss, C., Carter, J., Joyce, D., Harding, T., & Laxer, C. (2002). Addressing student cheating: definitions and solutions. ACM SigCSE Bulletin, 35(2), 172–184.

Joy, M., Cosma, G., Yau, J. Y. K., & Sinclair, J. (2010). Source code plagiarism—a student perspective. IEEE Transactions on Education, 54(1), 125–132.

Mann, S., & Frew, Z. (2006, January). Similarity and originality in code: plagiarism and normal variation in student assignments. In Proceedings of the 8th Australasian Conference on Computing Education-Volume 52 (pp. 143–150).

Mason, T., Gavrilovska, A., & Joyner, D. A. (2019, February). Collaboration versus cheating: Reducing code plagiarism in an online MS computer science program. In Proceedings of the 50th ACM Technical Symposium on Computer Science Education (pp. 1004–1010).

Sheard, J., Carbone, A., & Dick, M. (2003, January). Determination of factors which impact on IT students' propensity to cheat. In Proceedings of the fifth Australasian conference on Computing education-Volume 20 (pp. 119–126). Australian Computer Society, Inc.

Tartu Ülikooli õppekorralduseeskiri (31.01.2020). Kasutatud 12.04.2020, https://www.ut.ee/oke.

Walenstein, A., El-Ramly, M., Cordy, J. R., Evans, W. S., Mahdavi, K., Pizka, M., Ramalingam, G. & von Gudenberg, J. W. (2007). Similarity in programs. In Dagstuhl Seminar Proceedings. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

Wanhenheim, A. V., Martina, J. E., Cancian, R. L., & Dovichi, J. C. (2015). Developing Programming Courses with Moodle and VPL-The Teacher's Guide to the Virtual Programming Lab. Bookess.

Lisad

I Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Heli-Katri Marttila

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose
   Programmide sarnasuse määratlemine ühe programmeerimisülesande näitel

   mille juhendajad on

   Eno Tõnisson ja Heidi Meier

   reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

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

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

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

Heli-Katri Marttila

08.05.2020