TARTU ÜLIKOOL
Loodus- ja täppisteaduste valdkond
Matemaatika ja statistika instituut

Tiiu Kaljuste

Ülesannete kogu „Programmeerime endale Lahendaja“ näidislahendused

Matemaatika eriala

Bakalaureusetöö (9 EAP)

Juhendaja: Rein Prank, Emeriitdotsent

Tartu 2019

Ülesannete kogu „Programmeerime endale Lahendaja“ näidislahendused

Lühikokkuvõte:

Bakalaureusetöö eesmärk on koostada ülesannete kogule „Programmeerime endale Lahendaja“ näidislahendused. Töö teoreetilises osas antakse ülevaade põhikooli matemaatikakursusest tuntud algoritmide kasutamisest programmeerimise õpetamisel. Töö praktilises osas koostatakse ülesannete kogule näidislahendused aitamaks programmeerimise õpetajatel ja algajatel iseõppijatel leida ideid ülesannete lahendamiseks.

Võtmesõnad:

Informaatika, programmeerimine, ülesanded, põhikooli matemaatika algoritmid

CERCS: S281 Arvuti õpiprogrammide kasutamise metoodika ja pedagoogika

Example solutions for the “Let’s program ourselves a Solver” collection of exercises.

Abstract:

The aim of this paper is to create example solutions for the collection of programming problems “Let’s program ourselves a Solver”. The theoretical part gives an overview of the usage of algorithms, known from basic school mathematics, in teaching programming. In the practical part, example solutions are created for the aforementioned collection of programming problems to help the teachers of programming and beginners learning by themselves find ideas how to solve the programming problems.

Keywords:

Informatics, programming, problems, algorithms in basic school mathematics

CERCS: S281 Computer-assisted education

Sisukord

Sissejuhatus

Eestil on juba aastaid hea maine lastele programmeerimise õpetamises. Kahjuks ei ole hea maine taga nii laia kasvulava kui soovitakse. Infotehnoloogia spetsialistidest on nii Eestis kui ka kogu maailmas suur puudus. Ülikoolidesse võetakse infotehnoloogia erialadele vastu üha rohkem tudengeid. Et õpilastest kasvaks head programmeerijad, tuleb programmeerimise õpetamisele suuremat tähelepanu pöörata juba gümnaasiumis ja tutvustada programmeerimist põhikoolis. See omakorda toob kaasa suurema nõudluse õppematerjalide loomisele. Suurema õppematerjali vajadusega käib kaasas vajadus programmeerimisülesannete järele. On olemas vastavad ülesannete kogud juba kogenud õpilastele, kes kvalifitseeruvad programmeerimisolümpiaadidest osavõtjateks. Aga olümpiaadidele jõudmiseks on vaja ka algajatele jõukohasemaid ülesandeid. Samuti vajavad ülesandeid ringide ja kursuste läbiviijad. Miks mitte ka päris iseõppijad. Ka neile on vaja motiveerivaid ülesandeid.

Üht sellist kogu koostab praegu loodus- ja täppisteaduste emeriitdotsent Rein Prank. Tema koostatava ülesannete kogu teemad on üles ehitatud põhikooli matemaatika õpikute ülesannete põhjal. Selline kontseptsioon on kasulik mitmel tasandil. Põhikooli matemaatika on oma olemuselt algoritmiline, seega on väga hea selle põhjal programme koostada. Lisaks programmeerimisülesannete lahendamisele saab õpilasel ka matemaatika selgemaks. Ülesande lahendamiseks tuleb tal ülesandesse „sisse minna“ ja see annab talle hea põhja matemaatika mõistmiseks.

Antud töö eesmärgiks on koostatava „Programmeerime endale Lahendaja“ (2019) programmeerimisülesannete kogus [1] toodud ülesannetele lisada näidislahendusi.

Töö esimeses peatükis on ülevaade [1] ülesannete koguga analoogilistest teistest allikatest. Teises peatükis kirjutatakse programmeerimisülesannete esinemisest Eestis kasutatavates üldhariduslike koolide matemaatika õpikutes. Kolmandas peatükis vaadeldakse matemaatikateemaliste ülesannete kasutamist Eestis toimuvatel programmeerimise olümpiaadidel. Neljandas peatükis on juttu põhikooli matemaatika kursuses esitatavatest ülesande lahendamise algoritmidest. Viimases, viiendas osas on koostatava ülesannete kogu „Programmeerime endale Lahendaja ” näidisülesanded ja nende ülesannete selgitused.

1. Analoogilistest allikatest Eestis.

Programmeerimisülesannete kogusid on koostatud mitmesugustel põhjustel. Enamasti on need ülesannete kogud mõeldud tudengitele erinevate programmeerimiskeelte õppimiseks. Sellised ülesannete kogud on mingi kindla kursusega seotud ja ei ole kahjuks laiale avalikkusele kättesaadavad.

Üldhariduslikust matemaatikast tuttavaid ülesandeid on kasutatud Targo Tennisbergi ja Katrin Gabreli loodud „Võistlusprogrammeerimise õpikus“ [2]. Kuna see õpik on mõeldud peamiselt õpilastele, kes huvituvad just informaatika olümpiaadidest osavõtmisest, siis on ülesanded suunatud võimalikult optimaalseks lahendamiseks. Selles õpikus esinevate ülesannete lahendamisel kasutatakse ka juba vastavas programmeerimiskeeles valmiskujul olevaid alamprogrammide teeke. Seega on algajatel programmeerijatel, kes on läbinud esmase tehnilise programmeerimiskeele kursuse keerulisem programmide tööpõhimõttest aru saada. Edasijõudnud programmeerijatele on „Võistlusprogrammeerimise õpik“ kindlasti hea abimaterjal.

Nimetatud õpiku arvuteooria peatükis on põhikooli matemaatika kursusest hästi tuttavad suurima ühisteguri ja vähima ühiskordse ülesanded. Õpikust leiab selgitusi algarvude leidmiseks ja Eratosthenese sõela kasutamiseks. Samuti leiab sellest õpikust ülesandeid suurte arvudega tehete sooritamiseks. Õpiku 2. osas [3] on käsitletud nii põhikooli kui ka gümnaasiumi geomeetriat. Uuritakse lõikude lõikumise ülesandeid ja paralleelsuse kontrollimist. Kirjeldatakse ka hulknurkade pindalade leidmise võimalusi.

R. Prangi koostatava ülesannete kogu ülesannete lahendamise järel on kasulik tutvuda analoogsete ülesannete näidislahendustega teistes sarnase suunitlusega ülesannete kogudest.

2. Programmeerimisülesannete esinemisest Eestis kasutatavates matemaatika õpikutes

Paljusid matemaatika ülesandeid on hea lahendada arvuti abil. Enamasti on need suuremahulisi arvutusi sisaldavad ülesanded. Aga miks mitte ka vähest arvutamist vajavaid ülesandeid "õpetada" arvutiprogrammil lahendama. Kui õpilane peab kirjutama programmi, mis täidab kõiki ülesande lahenduse jaoks vajaminevaid reegleid, siis saab ta kindlasti ise ka need reeglid selgeks.

Tõnu Tõnso ja Allar Veelmaa poolt välja antud gümnaasiumile mõeldud matemaatika õpikutes "Matemaatika X klassile" [4], "Matemaatika XI klassile" [5] ja "Matemaatika XII klassile" [6] on mõningate ülesannete lahendamine ära kirjeldatud BASIC-keelse programmiga. Õpilane peab mitte ainult need programmid arvutisse sisestama vaid ka kirjeldama programmi töö põhimõtteid. Mõne programmi juures peab õpilane oskama ka ise BASIC-programmi käske muuta, et programm teeks vastavalt juhendile midagi muud.

X klassi matemaatikaõpikus on üks peatükk "Lühike juhend tööks BASIC'us", milles selgitatakse lahti algajatele vajalikud mõisted ja funktsioonid.

Järgnevalt näiteid 10. klassi matemaatika õpikust [4].

2.1 Arvusüsteemid

Näide 1: Järgnev programm teisendab araabia numbrisüsteemi arve rooma numbrisüsteemi arvudeks. Õpilane saab selle programmi käivitamisel sisestada erinevaid arve. Et programmi mugavamalt käsitleda ergutatakse õpilast ise programmi muutma, et saaks sisestada ainult naturaalarve ja vältimaks igakord uue arvu sisestamiseks programmi taaskäivitama, soovitatakse teha vastavad muudatused.

```
10 PRINT"Sisesta arv araabia numbrites";:INPUT A
20 PRINT"See arv on rooma numbrisüsteemis";
30 READ A$, P
40 IF P=A THEN 90
50 IF P>A THEN 30
60 A=A – P
70 PRINT A$;
80 GOTO 40
90 PRINT A$
100 DATA M, 1000, CM, 900, D, 500, CD, 400, C, 100
110 DATA XC, 90, L, 50, XL, 40, X, 10, IX, 9, V, 5, IV, 4, I, 1
120 END
```

Näide 2: Programm, mis võimaldab teisendada arve ühest arvusüsteemist teise. Tingimuseks on siiski, et süsteemi alus ei oleks suurem kümnest. Programmi tööpõhimõte on ülesande juures ära toodud. Algselt teisendatakse arv A kümnendsüsteemi (N) ja seejärel teisendatakse arv soovitud arvusüsteemi.

Paremaks arusaamiseks töö põhimõttest on ülesandeks teha sellest programmist kaks lühemat programmi, millest üks teisendaks arve K-süsteemist kümnendsüsteemi ja teine teisendaks kümnendsüsteemi arve L-süsteemi.

Raskem ülesanne on muuta seda programmi teksti nii, et saaks kasutada ka arvusüsteemide teisendamiseks, mille alus on kümnest suurem (näiteks kuni 16. süsteemini)

```
10 INPUT „Mitmendsüsteemist“;K
20 INPUT „Mitmendsüsteemi“;L
30 INPUT „Sisesta arv“; A
40 U = A MOD 10
50 N= N+U*K^W
60 W = W+1
70 A = A\10
80 IF A>0 GOTO 40
90 B$=STR$(N MOD L)+B$
100 N=N\L
110 IF N>0 GOTO 90
120 PRINT B$
130 END
```

Järgnevalt näiteid 11. klassi matemaatika õpikust [5].

2.2 Teema: Polünoomi nullkohad. Polünoomi lahutamine teguriteks

Näide 3. Personaalarvutit kasutades on vaja leida polünoomi väärtus antud kohal kasutades selleks Horneri skeemi. Peab oskama kirjeldada algoritmi, mille järgi programm töötab. Kui õpilane oskab, siis tuleb koostada ka algoritmi blokkskeem. Programmi tööd tuleb kontrollida õpikus toodud varasemate näidete järgi.

```
10 INPUT „SISESTA POLÜNOOMI ASTE N“;N
20 INPUT „POLÜNOOMI VÄÄRTUSE ARVUTAN KOHAL X“;X
30 REM ** KORDAJATE SISESTAMISE ALGUS **
40 FOR I = 0 TO N
50 PRINT “SISESTA A(“;I;“)=“;
60 INPUT A(I)
70 NEXT I
80 REM ** KORDAJATE SISESTAMISE LÕPP **
90 Y=A(0)
100 FOR I = 1 TO N
110 Y = Y * X + A(I)
120 NEXT I
130 PRINT “POLÜNOOMI VÄÄRTUS ON Y(“;X;“)=“;Y
140 END
```

2.3 Teema: Eksponent- ja logaritmfunktsioon

Näide 4. Arvu astmeid leidva programmi kirjutamine. Programm, mis arvutab arvu 2 astmeid, kui astendaja muutub 2-st 3-ni sammuga 0,1.

```
10 FOR X=2 TO 3 STEP 0.1
20 PRINT X; 2^X
30 NEXT
```

Näide 5. Programm trükib välja iga aasta lõpus arvel oleva rahasumma (kui arve avati aasta algul) ning ka intressina lisandunud summa.

```
10 INPUT “SISESTA ALGSUMMA“;A
20 INPUT “SISESTA INTRESSIMÄÄR PROTSENTIDES“;P
30 INPUT “SISESTA AASTATE ARV“;N
40 PRINT “ALGSUMMA ON“;A
50 FOR I=1 TO N
60 PRINT I;“. AASTA INTRESS ON“;A*P/100
70 A=A+A*P/100
80 PRINT “SUMMA “;I;“. AASTA LÕPUKS ON “;A
90 NEXT I
```

2.4 Teema: jadad

Näide 6. Programm leiab Fibonacci arve.

```
10 REM Fibonacci arvud
20 INPUT "Mitu arvu "; P
30 DIM A(P)
40 A(I)=1
50 A(2)=2
60 FOR N=1 TO P – 2
70 A(N + 2) = A(N + 1) + A(N)
80 PRINT A(N + 2)
90 NEXT N
100 END
```

2.5 Teema: kombinatoorika

Näide 7. Permutatsioonid. Siinolev programm võimaldab leida ka negatiivsete ja murdarvude „faktoriaale". Loomulikult pole see tegelikult võimalik, aga selleks peab programmi täiustama.

```
10 PRINT "PERMUTATSIOONID N ELEMENDIST"
20 INPUT "N=";N
30 P = 1
40 FOR I = 1 TO N
50 P = P*I
60 NEXT I
70 PRINT "P =";P
80 END
```

Näide 8. Kombinatsioonid. On vaja selgitada, mida programm teeb. Vajaduse korral lihtsustada või täiustada.

```
10 INPUT " n= ";N
20 INPUT " k= ";K
30 S=N:GOSUB 100
40 PN=T        'PN on arvu n faktoriaal
50 S=N – K: GOSUB 100
60 PNK = T     'PNK on arvu n-k faktoriaal
70 S=K: GOSUB 100
80 PK=T        'PK on arvu k faktoriaal
90 PRINT "variatsioone "; PN/PNK;" kombinatsioone: "; PN/(PNK*PK)
95 GOTO 150
100 T=1        'Faktoriaali arvutamise alamprogrammi algus
110 FOR I = 1 TO S
120 T = T * I
130 NEXT I
140 RETURN             'Faktoriaali arvutamise alamprogrammi lõpp
150 END
```

Järgnevalt programme XII klassi õpikust [6].

2.6 Teema: Statistika

Näide 9. Diskreetse juhusliku suuruse keskväärtuse ja standardhälbe arvutamise programm.

```
10 INPUT „Mitu erinevat tunnuse väärtust" , K
20 DIM X(K), Y(K)
30 KESKV = 0
40 FOR I = 1 TO K
50 PRINT „Sisesta X("; I ; „) "; : INPUT X(I)
60 PRINT „Sisesta P("; I ; „) "; : INPUT P(I)
70 KESKV = KESKV + X(I) * P(I)
80 NEXT I
90 PRINT „Keskväärtus on "; KESKV
```

```
100 DISP = 0
110 FOR I = 1 TO K
120 DISP = DISP + (X(I) – KESKV) ^2 * P(I)
130 NEXT I
140 PRINT „Standardhälve on “; SQR(DISP)
150 END
```

2.7 Teema: Integraal

Näide 10. Arvutiprogramm, mis leiab ristkülikuvalemi järgi määratud integraali funktsioonist $y=x^2$.

```
10 DEF FNK (X) = X ^2
20 INPUT “Alumine raja”, A
30 INPUT “Ülemine raja”, B
40 INPUT “Mitmeks osaks jaotada”, N
50 H = (B – A) / N
60 X = A + H / 2
70 SUMMA = 0
80 FOR I = 1 TO N
90 SUMMA = SUMMA + FNK(X)
100 X = X + H
110 NEXT I
120 S = H * SUMMA
130 PRINT S
140 END
```

Seda programmi saab kasutada ka teiste funktsioonide tarbeks. Kümnendale reale tuleb kirjutada vastav funktsioon.

3. Matemaatikateemaliste ülesannete kasutamisest Eestis toimuvatel programmeerimise olümpiaadidel

Eesti riiklikud informaatikaolümpiaadid on toimunud juba üle 25 aasta. Lisaks toimub ka mitmeid programmeerimisvõistlusi. Järgnevalt väike ülevaade, milliseid põhikooli matemaatikaülesannetel põhinevaid olümpiaadiülesandeid on esinenud. Mitmed alljärgnevatest ülesannetest on osaliselt või tervenisti ära toodud koostatavas ülesannete kogus. R. Prangi ülesannete kogus [1] on ülesanded „Tehtemärgid“, „Murdude lahutamine“ ja „Võrdus“.

3.1 Pindala. (Eesti informaatikaolümpiaad. Eelvoor 06.12.2014. [7])

Ruudulisele paberile saab joonistada kinniseid hulknurki, järgides ainult ruudustiku jooni. See tähendab, et kõik hulknurga küljed on horisontaalsed või vertikaalsed ning täisarvuliste pikkustega. Iga hulknurga joonistamise eeskiri on antud sõnena üksikute lõikude kaupa: W — vasakule, N — üles, E — paremale, S — alla. On teada, et hulknurk ei puutu ega lõika iseennast, s.t iga punkt hulknurga kirjelduses esineb ainult uks kord. Hulknurk on ka ortogonaalselt kumer. See tähendab, et iga horisontaalne või vertikaalne sirge, mis hulknurka lõikab, siseneb sellesse ja väljub sellest ainult ühe korra. Lihtsustatult, hulknurk ei sisalda näiteks U-kujulisi osi. Näiteks NNWSWSEE (*joonisel 1* vasakul) annab ortogonaalselt kumera hulknurga, aga SSEEENNWSWNW (joonisel paremal) mitte.

Joonis 1

Leida selliselt antud hulknurga pindala.

Sisend. Tekstifailis on täpselt kaks rida. Esimesel real on lõikude arv K. Teisel real on sõne pikkusega K, mis koosneb märkidest N, E, S ja W.

3.2 Arvuruut (Gümnaasium). (Eesti informaatikaolümpiaad. Lõppvoor 14.02.2015 [8])

Arvuruut on mäng, kus tuleb arvud 1...N paigutada N×N ruudustikku nii, et iga arv esineks igas reas ja igas veerus täpselt üks kord. Lisaks on mõnede kõrvuti asetsevate ruutude kohta teada, kummas ruudus peab olema suurem arv.

Sisendfailis on ette antud mõnedes ruutudes olevad arvud ja mõnede ruutude vahelised seosed. Täita ruudustik neid etteantud tingimusi arvestades.

3.3 Tehtemärgid. (Eesti informaatikaolümpiaad. Lahtine võistlus 12–18.10.2015. [9])

Antud jada positiivseid täisarve ning positiivne “vastus”. Panna arvude vahele pluss- ja miinusmärgid nii, et avaldise väärtus oleks etteantud vastus.

See ülesanne on ka R. Prangi koostatavas ülesannete kogus „Programmeerime endale Lahendaja“ [1].

3.4 Sarnased hulknurgad (gümnaasium). (Eesti informaatikaolümpiaad. Lahtine võistlus. Eelvoor 19.11.2016. [10])

Juku õpib koolis hulknurkade sarnasust ja saab teada, et hulknurgad on sarnased, kui nende vastavate nurkade suurused on võrdsed ja vastavate külgede pikkused võrdelised. Sarnased hulknurgad võivad olla omavahel pööratud, peegeldatud ja nihutatud. Sarnaste hulknurkade vastavate külgede pikkuste jagatist nimetatakse nende sarnasusteguriks.

Kodutööna saab ta hulga hulknurki, mille sarnasustegureid on vaja määrata. Jukul on fanaatiline matemaatikaõpetaja, kes andis tööna väga paljude nurkadega hulknurki. Aita ta hädast välja.

3.5 Sarnased kolmnurgad (põhikool). (Eesti informaatikaolümpiaad. Lahtine võistlus. Eelvoor 19.11.2016. [10])

Juku õpib koolis kolmnurkade sarnasust ja saab teada, et kolmnurgad on sarnased, kui nende vastavate nurkade suurused on võrdsed ja vastavate külgede pikkused võrdelised. Sarnased kolmnurgad võivad olla omavahel pööratud, peegeldatud ja nihutatud. Sarnaste kolmnurkade vastavate külgede pikkuste jagatist nimetatakse nende sarnasusteguriks.

Kodutööna saab ta hulga kolmnurki, mille sarnasustegureid on vaja määrata. Kuna 21. sajandil sobivad arvutamiseks rohkem arvutid kui inimesed, aita Jukul kirjutada programm, mis selle töö tema eest ära teeb.

3.6 Murdude lahutamine. (Eesti informaatikaolümpiaad Eelvoor 18.11.2017. [11])

Kirjutada programm, mis oskab harilikke murde lahutada.

Sisend. Tekstifaili murdsis.txt esimesel ja teisel real on kaldkriipsu abil kirjutatud murrud a/b ja c/d, kus a ja c on mittenegatiivsed ning b ja d positiivsed täisarvud suurusega kuni 1000. Leida vahe a/b − c/d lihtmurru või segaarvuna, kus murd on taandatud. Väljastada tulemus formaatimata kujul ja formaadituna.

3.7 Kunstinäitus. (Eesti informaatikaolümpiaad Eelvoor 08.12.2018. [12])

Igal aastal kogunevad paljude maade noored programmeerijad, et osaleda Eriti Informaatilisel Olümpiaadil. Lisaks võistlusele külastatakse ka muuseume ja muid vaatamisväärsusi. Tänavu korraldatakse olümpiaadil osalejatele spetsiaalne kunstinäitus ja sihtgrupile meeldimiseks koosneb iga pilt hulgast täisarvuliste koordinaatidega punktidest. Igal pildil olevate punktide koordinaadid on juba otsustatud ja nüüd on jäänud veel pildid välja trükkida. Nende tavalisele ristkülikulisele lõuendile trükkimine võib aga asjata materjali raisata, sest suurel osal pinnast ei tarvitse siis ühtegi punkti olla. Loodusressursside säästmiseks otsustati, et iga lõuend peab olema nelinurk, mille ülemine ja alumine serv on rangelt horisontaalsed, mis sisaldab kõiki punkte ja mille pindala on seejuures vähim võimalik. Kirjutada programm, mis leiab lõuendi minimaalse pindala. Seejuures võib lõuendi iga serv olla kuitahes lühike. Muuhulgas võib lõuend kiduda kolmnurgaks, jooneks või isegi punktiks, ja kahel viimasel juhul on tema pindala null.

3.8 Võrdus. (Eesti informaatikaolümpiaad. Eelvoor 08.12.2018. [12])

Paigutada nelja antud täisarvu vahele märgid +, -, * või = nii, et tekiks tõene võrdus.

3.9 Keskmine hinne. (Eesti informaatikaolümpiaad. Harjutusvoor 10.11.2018. [13])

Kirjutada programm, mis saab riigi õpilaste nimekirja ja sorteerib selle keskmise hinde järgi.

3.10 Marsi kaart. (Balti informaatikaolümpiaad. 2001. [14])

Aastal 2051 on mitmed ekspeditsioonid uurinud erinevaid punase planeedi alasid ja koostanud nendest aladest kaardid. Nüüd on Baltic Space Agency-l ambitsioonikas plaan koostada terve planeedi kart. Et arvutada välja kogu eeldatav töö maht on vaja kokku lugeda juba olemasolevate kaartide pindala. Kirjutada programm, mis arvutab välja olemasolevate kaartide pindala.

Sisendis on kaartide arv ja iga kaardi kohta neli täisarvu x1, y1, x2, y2, mis vastavad kaardi vasak-alumise ja parem-ülemise punkti koordinaatidele. Kaardid on ristkülikukujulised.

3.11 Kolmnurgad. (Balti informaatikaolümpiaad. 2002. [15])

On antud võrdkülgsed täisnurksed kolmnurgad. Kõik kolmnurgad on kirjeldatud kolme täisarvulise muutujaga x, y, m (m>0). Kolmnurga tippude koordinaadid on (x; y), (x+m; y) ja (x; y+m). Kirjutada programm, mis arvutab kogu kolmnurkade poolt kaetud pindala. (Kolmnurgad võivad kattuda).

4. Matemaatika kursuses esitatavatest algoritmidest.

TÜ arvutiteaduse instituudi programmeerimisõpikus [16] selgitatakse algoritmi mõistet järgmiselt: **Algoritmiks** nimetatakse probleemi lahendamiseks vajalikku instruktsioonide hulka, mida *mehaaniliselt* (st ilma loovust rakendamata) järgides on võimalik jõuda soovitud tulemuseni. Algoritmi kohta öeldakse tihti ka lihtsalt *protseduur*.

Algoritmil on neli olulist omadust.

1. Algoritmi iga samm peab olema *täpne*, st olema ühetähenduslik.
2. Algoritm peab olema *lõplik*. Vastasel juhul ei saa me probleemile lahendust.
3. Algoritm peab olema *efektiivne*, st ta peab andma probleemile korrektse vastuse.
4. Algoritm peab olema *üldine*, st ta peab lahendama ülesande iga eksemplari. Näiteks ringi pindala leidmise algoritm peab sobima kõigi võimalike algandmetega.

Järgnevalt mõned näited põhikooli matemaatika õpikutest esinenud ülesannetest, millele on antud lahendamise algoritm.

Matemaatika III klassi õpikus[17] on kirjaliku lahutamise ülesanne selgitusega:

Ülesanne 680. Lahutame kirjalikult 465 – 124:

```
 465
–124
```

Esmalt lahutame ühelised:

```
 4 6 5
–1 2 4
-------
       1
```

1 kirjutame üheliste alla.

Siis lahutame kümnelised:

```
 4 6 5
–1 2 4
-------
     4 1
```

4 kirjutame kümneliste alla.

Lõpuks lahutame sajalised:

```
 4 6 5
–1 2 4
-------
 3 4 1
```

3 kirjutame sajaliste alla.

Edasi on selgitus ka ülesandele, kus lahutatava ühelistest, kümnelistest, jne., lahutaja vastavad järkarvud on suuremad lahutatava omast.

Ülesanne 689. Koolis on oktoobrilapsi ja pioneere kokku 470. Oktoobrilapsi on 128. Kui palju on koolis pioneere?

```
 4 7 0
–1 2 8
-------

 3 4 2
```

Seletus. Nullist ühelisest ei ole võimalik lahutada 8 ühelist. Seepärast võtame 7-st kümnelisest 1 kümnelise ja teeme ühelisteks. Kümneliste kohale paneme punkti. See näitab, et nüüd on üks kümneline vähem. [17]

Siin on kasutatud varasemalt selgitatud algoritmi kirjaliku lahutamise kohta ja lisatud algoritm juhuks, kui lahutatava vastav järk on väiksem kui lahutajal. Kui need algoritmi sammud üldistada on hea kirjutada arvutiprogramm.

Samast õpikust ka kirjaliku jagamise selgitus. Jagada järgemööda kõik järguühikute arvud antud jagajaga, alustades kõrgemast järgust, s.o. vasakult. Enne, kui jagamist alustame, teeme kindlaks, mitu numbrit saame jagatisse. Olgu vaja leida näiteks jagatis 3570:7.

Jagatava tuhandeline on väiksem kui jagaja 7. Seetõttu me jagatisse tuhandelisi ei saa. Jagatava 3 tuhandelist ja 5 sajalist sisaldavad kokku 35 sajalist. 35 on suurem kui 7, seega koosneb jagatis sajalistest, kümnelistest ja ühelistest. Märgime nende numbrite kohad jagatises punktidena:

3570 : 7 = . . .	Arvutame:	Kirjutame:
	35 sajalist : 7 = 5 sajalist	5
	7 kümnelist : 7 = 1 kümneline	1
	0 ühelist : 7 = 0 ühelist	0

Vastus: 3570 : 7 = 510

Siit edasi minnes:

Ülesanne 975. Leiame kirjalikult 975 : 3.

975 : 3 = 325 15 0	Arvutame:	Kirjutame võrdusmärgi järele:
	9 : 3 = 3	3
	7 : 3, jääk 1	2, jagatava alla tõmbame joone, selle alla kirjutame jäägi 1. Järgmine osajagatav on 10 + 5 = 15
	15 : 3 = 5	5

Siit edasi saab samasuguse algoritmi järgi arvutada ka jagatist kahekohalise jagajaga:

Ülesanne117. Aastas on 12 kuud. Mitu aastat on 324 kuud?

Ülesande lahendamiseks tuleb kuude arv 324 jagada aasta kuude arvu 12-ga:

324 : 12 = 27

24

84

84

0

Vastus. 27 aastat.

Jagatava sajaliste arv on väiksem jagajast, seega jagatises sajalisi ei ole. Jagatava sajalistes ja kümnelistes on kümneid kokku 32; 32 on suurem, kui jagaja, seega jagatises on kümnelisi. Neid on 2, sest 3 * 12 > 32, kuid 2 * 12 < 32, jagatise esimeseks (kümneliste numbriks on seega 2. Nüüd selgitame, mitu kümnelist on jagatud: korrutame 2 * 12, saame 24. Selle lahutame 32-st, saame jäägi 8 kümnelist. Muudame jäägi 8 kümnelist ühelisteks ja

liidame sellega veel jagatavas olevad 4 ühelist. Saadud 84 ühelise jagamisel 12-ga saame jagatise üheliste arvuks 7. Korrutame: 7 * 12 = 84, kirjutame selle osajagatava 84 alla ja lahutame, saame jäägiks 0. [17]

Selliste näiteülesannete järgi on võimalik teha üldistusi ja lahendada kõiki sarnaseid ülesandeid.

Järgnevalt näide „Matemaatika IV klassile“[18] õpikust võrrandi ax + b = c lahendamiseks algoritm.

Vaatleme näiteks muutujat sisaldavat võrdust 2x + 245 = 397.

Sellise võrduse kohta ei saa öelda, kas ta on tõene või väär. Võib aga selgitada, missugune peab olema muutuja x väärtus, et võrdus annaks tõese lause. Muutujat x nimetatakse sel juhul otsitavaks ehk tundmatuks. Liidetava 2x leidmiseks tuleb summast 397 lahutada teine liidetav 245. Järelikult 2x = 397 – 245 ehk 2x = 152. Nüüd on antud kahe arvu korrutis (152) ja üks tegur on (2). Tundmatu teguri leidmiseks tuleb korrutis jagada antud teguriga. Nii saame, et x = 152 : 2 ehk x = 76.[18]

Selles näites on kasutatud varasemalt õpitud seoseid liitmise-lahutamise ja korrutamise-jagamise vahel.

Nagu näidete põhjal on näha, tuuakse matemaatika õppimise käigus algoritme sisse näiteülesannete abil. Alustatakse lihtsamatest juhtudest ja lisatakse sammhaaval uusi kuni saadakse lõpuks kokku üldine algoritm.

5. Ülesannete kogu “Programmeerime endale Lahendaja” näidislahendused

Ülesannete kogus on põhikooli I – VIII klassi matemaatika õpikutest pärinevad ülesanded, mis on hea materjal programmeerimise õppimiseks ja harjutamiseks, sest need ülesanded on algoritmilised ja seetõttu saab nende lahendamiseks kirjutada arvutiprogrammi. Ülesanded on jaotatud klasside kaupa. Kuid iga klassi ülesanded on koostatud nii, et on päris lihtsaid ülesandeid, mida on hea lahendada algajail programmeerijail ja kasutada ka programmeerimise õpetamisel. On ka keerulisi, olümpiaaditasemega ülesandeid.

Näidislahendused on kirjutatud programmeerimiskeeles Python. Paljudes programmeerimiskeeltes on tihti kasutatavate algoritmide teegid. Et õppida hästi programmeerima on antud ülesannete kogu ülesanded soovituslikult lahendamiseks madalama tasemega funktsioonide abil. Selliselt antud näidislahendused oleksid võimalikult arusaadavad ka teiste programmeerimiskeelte kasutajatele.

Ülesannete püstitustes ei ole alati nõutud algandmete ja lõpptulemuse formaadi kontrollimist, seega on programmeerija ülesandeks sisend- ja väljundformaat enne programmi kirjutamist ise fikseerida. Muidu oleks lihtsamate ülesannete juures algandmete formaadi kontrollimine mitmeid kordi keerulisem ülesandest endast. Samuti ei pea tähelepanu pöörama programmide ajamahukusele nagu olümpiaadide ülesannetes. Programmeerija saab rahulikult keskenduda vaid ülesande lahendamisele.

Näidislahenduste failide nimed on tuletatud ülesannete kogus toodud klasside ja ülesande numbrite järgi. Näiteks: I klassi ülesanne number 7 on võrdlusmärgi panemine tehte tulemuse ja arvu vahele. Järelikult on selle ülesande programmifail nimega (I-7) Võrdlusmärgi valimine.py ja temale vastav sisendfail on lihtsalt ülesande numbriga (I-7).txt tekstifail. Alates viienda klassi ülesannetest on faili nimele lisatud alateema tähis. Näiteks naturaalarvude liitmise ja lahutamise teema juures ülesande 1. „Naturaalarvu esitamine järkarvude summana“ näidislahenduse leiab failist (V-NL-1) Arvujärgud.py ja vastav sisendfail on nimega (V-NL-1).txt

Ülesannete kogus ei ole rangelt piiritletud sisendfailist loetavate andmete formaat. Seetõttu ei ole ka näidislahendustes suurt rõhku pandud andmete sisselugemise kontrollimisele. Eeldatakse, et andmed on korrektselt sisendfailis kirjas.

Järgnevalt on ära toodud näidislahenduste failid. Ülesande tekst, mis on võetud ülesannete kogust „Programmeerime endale Lahendaja“ [1] on iga ülesande juures kirjutatud kursiivis.

5.1 Näidislahendused

5.1.1 Ülesanne I-7. Võrdlusmärgi panemine tehte tulemuse ja arvu vahele

Antud: sõne kujul $a \oplus b \lessgtr c$ kus a, b ja c on naturaalarvud ning $\oplus$ on pluss või miinus.
Leida: sobiv võrdlusmärk $\lessgtr$ kohale (<, = või >).

Faili nimi: (I_7) Võrdlusmärgi valimine.py

```
#============== Programm ==============================
#Muutujad:
# avaldis - sisendfailist loetud sõne
# vasakpool - sõne, mis koosneb arvudest ja tehtemärkidest (+/-) nende vahel
# parempool - naturaalarv
# summa - vasakul pool olevate naturaalarvude summa
# arvud - vasakul pool olev sõne eraldatud täisarvudeks
#         lahutamistehe on teisendatud negatiivse arvu liitmiseks
#---------------------------------------------------------------------
#Sisendfailist avaldise lugemine
fail = open("(I_7).txt","r")
avaldis = fail.readline()
print("Avaldis ",end="")
print(avaldis)
fail.close()
#---------------------------------------------------------------------
summa = 0
# eraldame vasaku ja parema poole "?" kohalt
vasakpool, parempool = avaldis.split("?")
# teisendame lahutamistehte negatiivse arvu liitmiseks
vasakpool = vasakpool.replace("-", "+-")
if vasakpool[0] == "+": #kui esimesene arv on negatiivne
    vasakpool = vasakpool[1:]
parempool = int(parempool)
# eraldame vasaku poole avaldisest kõik arvud (koos "-"-märkidega
arvud = vasakpool.split("+")
for i in range(len(arvud)):
    summa += int(arvud[i])
if summa < parempool:
    print(vasakpool,"<",parempool)
elif summa > parempool:
    print(vasakpool,">",parempool)
else:
    print(vasakpool,"=",parempool)
#============== Programmi lõpp ==============================
```

Näide programmi töö tulemusest:

Sisend: -24+35?24

Väljund: -24+35<24

Näidislahenduses on kasutatud ideed asendada kõik lahutamistehted negatiivse arvu liitmistehtega ehk avaldise a – b korral saab avaldisest a + – b. Kuna esialgne avaldis on sõne-tüüpi, siis on hea kasutada funktsiooni sõne tükeldamiseks (split(„+“)) ja lihtsalt liita saadud tükid täisarvulistena kokku.

Ilmnes ka probleem. Kui vasaku poole avaldise esimene arv oli negatiivne, siis tükeldades jäi esimeseks tükiks tühi sõne („“) ja tühjast sõnest ei saa loomulikult täisarvu võtta. Probleemile sai lahenduseks kontrollida, kas esimesel positsioonil on pärast „+“-de lisamist märk „+“ (sest vaid negatiivse arvu korral lisab esimesele positsioonile „+“) ja jätta seejärel esimene sümbol vasakust poolest välja.

5.1.2 Ülesanne I-10 Muutujaga aditiivse avaldise väärtuse leidmine

Antud: *avaldis* $a_1 \oplus \ldots \oplus a_k$, *kus* $a_1, \ldots, a_k$ *on naturaalarvud või muutuja x , mille vahel on plussid ja miinused, ja muutuja x väärtus.*

Leida: *avaldise väärtus.*

Faili nimi: (I_10) Muutujaga avaldise väärtus.py

```
#============== Programm ==============================
# Muutujad:
# avaldis - sisendfailist loetud sõne
# x - sisendfailist loetud x väärtus (string)
# summa - sisendfailist loetud avaldise väärtus
# arvud - sisendfailist loetud avaldis eraldatud täisarvudeks,
#         lahutamistehe on teisendatud negatiivse arvu liitmiseks
#-------------------------------------------------------------
# Sisendfailist avaldise lugemine
fail = open("(I_10).txt","r")
avaldis = fail.readline()
loe_x = fail.readline()
fail.close()
x = int(loe_x)
print("Antud avaldis ",avaldis," kus x=",x)
fail.close()
#-------------------------------------------------------------------------
summa = 0
# teisendame lahutamistehte negatiivse arvu liitmiseks
avaldis = avaldis.replace("-", "+-")
if avaldis[0] == "+":
    avaldis = avaldis[1:]
arvud = avaldis.split("+")
for i in range(len(arvud)):
    if arvud[i] == 'x':
        summa += x
    else:
        summa += int(arvud[i])
print("Antud avaldise väärtus on ",summa)
#============== Programmi lõpp ==============================
```

Näide programmi töö tulemusest:

Sisend:

```
-25-3+34+x-34-45+56-58-25
-7
```

Väljund:

```
Antud avaldis  -25-3+34+x-34-45+56-58-25
  kus x= -7
Antud avaldise väärtus on  -107
```

Ülesande algoritmi idee on sarnane eelmise ülesande algoritmile – vahetada lahutamiste-he negatiivse arvu liitmistehteks. Edasi kontrollitakse, kas liidetav on „x“. Kui jah, siis tuleb liita juurde antud x-väärtus. Kui ei ole, liidab vastava arvu.

5.1.3 Ülesanne II-3 Kellaajale minutite liitmine

Antud: *kellaaeg (tunnid ja minutid) ja ooteaeg a (minutites).*

Leida: *kellaaeg a minuti pärast.*

Faili nimi: (II-3) Kellaaeg+minutid.py

```
#============== Programm ==============================
# Muutujad:
#    kellaaeg - andmefaili esimeselt realt sisseloetud kellaaeg
#    ooteaeg - andmefaili teiselt realt sisse loetud lisatavad minutid
#    tunnid, minutid - algne kellaaeg jaotatud tundideks ja minutiteks
#    juurde_minut - algsed minutid ja juurdelisatud minutid koos
#---------------------------------------------------------------
# Sisendfailist avaldise lugemine
fail = open("(II-3).txt","r")
kellaaeg = fail.readline()
print("Algne kellaaeg", kellaaeg,end=" ")
ooteaeg = fail.readline()
ooteaeg = int(ooteaeg)
print("Ooteaeg", ooteaeg)
fail.close()
tunnid, minutid = kellaaeg.split(".")
#---------------------------------------------------------------
juurde_minut = int(minutid) + ooteaeg
tunnid = int(tunnid) + int(juurde_minut/60)
# % annab jagatise jäägi
minutid = juurde_minut % 60
# alates 24-st tunnist hakkab lugemine jälle 0-st.
# välja arvatud kellaaeg 24.00
if (tunnid > 23) and (minutid > 0):
    tunnid = tunnid % 24
# ka 0 ... 9 minutit on vaja kirjutada kahekohalistena
minutid = str(minutid)
if len(minutid) < 2:
    minutid = "0" + minutid
print("Lõppaeg "+str(tunnid)+"."+minutid)
#============== Programmi lõpp ==============================
```

Näide programmi töö tulemusest:

Sisend:

```
23.04
76
```

Väljund:

```
Algne kellaaeg 23.04
 Ooteaeg 76
Lõppaeg 0.20
```

Tähelepanu tuleb pöörata kellaaja korrektsele väljastamisele. Minutid on vaja kirjutada kahekohalistena ka siis, kui minuteid on vaid 4 või 7. Samamoodi vajab erilist tähelepanu kellaaeg üle 24. Siin toodud näidislahenduses on jäetud kellaaeg 24.00 ja edasi on juba kellaaeg 0.01.

5.1.4 Ülesanne III-5 Võluruudu kontroll

Antud:

a) 3x3 naturaalarvu,

b) 4x4 naturaalarvu,

c) 5x5 naturaalarvu.

Leida: *kas arvud moodustavad võluruudu.*

a) ridade ja veergude summad on võrdsed,

b) ridade, veergude ja diagonaalide summad on võrdsed.

Faili nimi: (III-5) Võluruudu kontroll.py

```
#============== Programm ==============================
# Muutujad:
# ruut - nimekirja tüüpi muutuja sisseloetud andmetega, kus peaks olema read == veerud
# on_ruut - tõeväärtus, kas on üldse tegemist ruuduga
# kokku_ruut - hulga tüüpi muutuja, milles hoitakse ridade ja veergude summat. Pärast ridade või veergude
# summa lisamist hulka peab jääma vaid üks liige, sest võrdseid summasid ei lisata. Vastupidisel juhul
# annab teate mittesobimisest.
# summa, summa1 ja summa2 - ridade, veergude, diagonaalide summad
fail = open("(III-5).txt","r")
# loeb sisse sisendfailist andmerea ilma reavahetuse märgendita
ruut = fail.read().splitlines()
for i in range(len(ruut)):
    print(ruut[i])
for i in range(len(ruut)):
    ruut[i] = ruut[i].strip(" ") # lõikab ära kõik rea lõppu
    # jäänud tühikud, mis on andmeid sisestades kogemata tekkinud
    ruut[i] = ruut[i].split(" ")
    for j in range(len(ruut[i])):
        ruut[i][j] = int(ruut[i][j])
fail.close()

# kontrollib, kas on üldse ruut ja read-veerud ei ületaks
# ülesande tingimusi
on_ruut = True
for i in range(len(ruut)):
    if (not len(ruut[i]) == len(ruut)) or (len(ruut) >5):
        on_ruut = False
        print("Algandmed ei vasta ülesande tingimustele")
```

```
        break # ülesande tingimus pole täidetud, jätab kontrollimise
katki

if on_ruut:
    # variant a) kas ridade ja veergude summad on võrdsed
    kokku_summa = set()
    kontroll = True
    for i in range(len(ruut)):
        summa = 0
        for j in range(len(ruut[i])):
            summa += ruut[i][j]
        kokku_summa.add(summa)
        if len(kokku_summa) > 1:
            kontroll = False
            print("read ei ole võrdse summaga")
            break # tingimus pole täidetud, jätab kontrollimise katki

    kokku_summa = set()
    for j in range(len(ruut)):
        summa = 0
        for i in range(len(ruut[i])):
            summa += ruut[i][j]
        kokku_summa.add(summa)
        if len(kokku_summa) > 1:
            kontroll = False
            print("veerud ei ole võrdse summaga")
            break # tingimus pole täidetud, jätab kontrollimise katki

    # variant b) vaatame, kas diagonaalid on võrdsed
    summa1, summa2 = 0,0
    kokku_summa = set()
    for i in range(len(ruut)):
        summa1 += ruut[i][i] #diagonaal ülevalt vasakult
        summa2 += ruut[i][len(ruut)-i-1] #diagonaal ülevalt paremalt
    kokku_summa.add(summa1)
    kokku_summa.add(summa2)
    if len(kokku_summa) > 1:
        kontroll = False
        print("diagonaalid ei ole võrdse summaga")
    if kontroll:
        print("Tegemist on võluruuduga")
    else:
        print("Tegemist ei ole võluruuduga")
#============== Programmi lõpp ==============================
```

Näited programmi töö tulemustest:

1) Väljund:

```
1 2 3 4
2 3 4 5
3 4 5 6
Algandmed ei vasta ülesande tingimustele
```

2) Väljund:

```
1 3 2
3 2 1
2 1 3
Tegemist on võluruuduga
```

3) Väljund:

```
1 2 3 4 5
5 1 2 3 4
4 5 1 2 3
3 4 5 1 2
2 3 4 5 1
diagonaalid ei ole võrdse summaga
Tegemist ei ole võluruuduga
```

5.1.5 Ülesanne IV-2 Suuruselt teine arv

Leida arvujada suuruselt teine arv.

Sisend: *Sisendfaili real pikkusega kuni 80 sümbolit on tühikuga eraldatud naturaalarvud.*

Väljun:. *Suuruselt teine arv arvujadas.*

Faili nimi: (IV-2) Suuruselt teine arv.py

```
#============== Programm ==============================
# Muutujad
#    loe_rida - andmefailist 80-sümboliline rida
#    arvud - nimekiri sisseloetud rea naturaalarvudest
#    esimene - suuruselt esimene ar
#    teine - suuruselt teine arv
#---------------------------------------------
fail = open("(IV-2).txt","r")
# loeb failist vaid 80 esimest sümbolit. Failis võib neid olla rohkem
loe_rida = fail.readline(80)
fail.close()
loe_rida = loe_rida.rstrip(" ") #lõikab ära paremal olevad tühikud
print("Failist loetud arvud",loe_rida)
arvud = loe_rida.split(" ")
#kui sisendreas on arvude vahel kogemata rohkem tühikuid, jätab välja
i = 0
while i < len(arvud):
    if arvud[i] == "":
        arvud.remove(arvud[i])
    else:
        i += 1

# määrab algseisuks esimesest kahest liikmest suurima esimeseks
esimene = max(int(arvud[0]),int(arvud[1]))
teine = min(int(arvud[0]),int(arvud[1]))
for i in range(2,len(arvud)):
    # võrdleb hetkel teise tulemusega
    if int(arvud[i]) > teine:
        # kui oli teisest suurem, tuleb vaadata, kas ka esimesest
        if int(arvud[i]) > esimene:
            teine = esimene # peab enne tegema, sest võrreldakse esime-
sega
            esimene = int(arvud[i])
        else:
            teine = int(arvud[i])
print("Suuruselt teine arv reas on",teine)
```

```
#============== Programmi lõpp ==============================
```

Näide programmi töö tulemusest
Sisend: 11 2 3 4 -15 6 07 5 11 2 3 4 15 -6 07 7 31 2 3 4 -15 776 07 8 11 2 1233 4 -15 6 67 88 -888
Väljund:
Failist loetud arvud 11 2 3 4 -15 6 07 5 11 2 3 4 15 -6 07 7 31 2 3 4 -15 776 07 8 11 2 1233 4 -15 6
Suuruselt teine arv reas on 776

Tähelepanu tuleb pöörata ülesande tingimusele lugeda vaid 80 esimest sümbolit. Kõik ülejäänud jäävad arvestusest välja.

5.1.6 Ülesanne V-NL-1 Naturaalarvu esitamine järkarvude summana

Antud: *positiivne naturaalarv a.*

Esitada arv a järkarvude summana (alates sobivast järgust).

Sisend: *Sisendfaili ainsal real on naturaalarv a, kus 0 < a ≤ 1000000.*

Väljund: *Kirjutada väljundfaili esimesele reale võrdus a =< järkarvude summa >.*

Faili nimi: (V-NL-1) Arvujärgud.py

```
#============== Programm ==============================
# Naturaalarvu esitamine järkarvude summana
# Sisendiks on positiivne naturaalarv
#-------------------------------------------------------------
# avaldis - sisendfailist sisseloetav positiivne täisarv
#------------------ Programmi algus ----------------------
fail = open("(V-NL-1).txt","r")
avaldis = fail.readline()
fail.close()
print(avaldis + "=",end=““)
for i in range(0,len(avaldis)-1,1):
    if avaldis[i] != '0':
        print(int(avaldis[i])*10**(len(avaldis)-i-1), end='+')
print(avaldis[len(avaldis)-1])
#============== Programmi lõpp ============================
```

Näide programme töö tulemusest

Sisend: 654321

Väljund: 654321=600000+50000+4000+300+20+1

5.1.7 Ülesanne V-NK-3 Kirjalik korrutamine

Antud: *kaks kuni 5-kohalist naturaalarvu.*

Esitada nende korrutamine tavalise kirjaliku algoritmi järgi, paigutades arvud õigetele positsioonidele.

Faili nimi: (V-NK-3) Kirjalik korrutamine.py

```
#============== Programm ==============================
# Muutujad:
#     avaldis - failist loetav sõne, kus on kaks arvu
#     arvud - sõnest erldatud kaks täisarvu
#     korrutis - kahe arvu korrutis
#     lpikkus - lõppkorrutise sümbolite arv
#     vpikkus - säilitab lpikkuse väärtuse vahejoone pikkuse
#     määramiseks
#     vahekorrutis = iga rea ehk iga 2.teguri järgu korrutis
#-------------------------------------------------------
fail = open("(V-NK-3).txt","r")
avaldis = fail.readline()
fail.close()
arvud = list(avaldis.split(' '))
korrutis = int(arvud[0]) * int(arvud[1])
lpikkus = len(str(korrutis))

# lisab tühikuid, et kirjutis saaks kohakuti
print(str(arvud[0]).rjust(lpikkus))
print(str(arvud[1]).rjust(lpikkus))
print('%s'%'-'* lpikkus)

pikkus=len(avaldis[1])
arv=list(arvud[1])

#pöörab muutuja arv sümbolid ümber. Kui enne on "123",
# siis pärast "321", sest korrutama hakkame tagant
# ettepoole
arv.reverse()
vpikkus = lpikkus
for i in arv:
    if not i == '0':
        vahekorrutis = int(arvud[0])*int(i)
        print(str(vahekorrutis).rjust(vpikkus))
    vpikkus -= 1
print('%s'%'-'* lpikkus)
```

```
print(korrutis)
#============== Programmi lõpp ==============================
```

Näide programmi töö tulemusest:

Väljund:

```
     20874
     86543
----------
     62622
    83496
  104370
 125244
166992
----------
1806498582
```

5.1.8 Ülesanne V-NK-5 Võrrandi ax=b lahendamine

Antud: Võrrand kujul ax = b, kus a ja b on täisarvud.

Leida: Võrrandi lahend.

Sisend: Sisendifaili ainsal real on võrrand ax = b, kus aja b on täisarvud ja x tundmatu, kusjuures arv b jagub a-ga täpselt.

Faili nimi: (V-NK-5) ax=b lahendamine.py

```
#============== Programm ==============================
# Muutujad:
#    avaldis - sisendfailist loeb avaldise arvA x = arvB
#    arvA
#    arvB
#    x - tundmatu, mida otsime
#------------------------------------------------------------
fail = open("(V-NK-5).txt","r")
avaldis = fail.readline()
fail.close()
print(avaldis)
arvA,arvB = avaldis.split("=")

arvA,x = arvA.split("x")
x = int(arvB)/int(arvA)

if not x == int(x):
    print("Arv", arvB,"ei jagu täpselt arvuga",arvA,", seega ülesande tingimus pole täidetud")
else:
    print("x = ",int(x) )
#============== Programmi lõpp ==============================
```

Näited programme töö tulemustest:

1) Väljund:

```
250x=-100
Arv -100 ei jagu täpselt arvuga 250 , seega ülesande tingimus pole täidetud
```

2) Väljund:

```
-15x=765
x =  -51
```

5.1.9 Ülesanne V-NK-10 Tegurite leidmine

Antud: *positiivne naturaalarv a.*

Leida: *arvu a kõik tegurid.*

Faili nimi: (V-NK-10) Tegurite leidmine.py

```
#============== Programm =============================
#Muutujad
#    a - sisendfailist sisseloetav positiivne naturaalarv
#    tegurid - nimekiri arvu a teguritest
#    jagatis - kui jagatis on täisarv, siis lisab nimekirja
#-------------------------------------------------------------
# Ruutjuure leidmiseks on vaja kasutada lisaks
# matemaatika moodulit
from math import *
fail = open("(V-NK-10).txt","r")
a = int(fail.readline())
fail.close()
tegurid = [1]
if a > 1:
    tegurid.append(a)

for i in range(2,int(sqrt(a)+1)): #nulliga ei tohi, 1 on nagunii
# sqrt() võtab ruutjuure, kaugemalt pole mõtet otsida
    jagatis = a/i
    # lisab nimekirja leitud arvud, mis täpselt jaguvad
    if jagatis == int(jagatis):
        tegurid.append(i)
        if not jagatis in tegurid:
            tegurid.append(int(jagatis))
tegurid.sort()   #väljund korrektselt järjekorras
print("Arvu "+str(a)+" tegurid on:",end=" ")
for i in range(len(tegurid)):
    print(tegurid[i],end=", ")
#============== Programmi lõpp ==============================
```

Näiteid programme töö tulemusest:

Väljund 1: `Arvu 2 tegurid on 1, 2`

Väljund 2: `Arvu 6534565 tegurid on: 1, 5, 1306913, 6534565,`

5.1.10 Ülesanne V-NK-13 Eratosthenese sõel

Antud: naturaalarv N>1

Leida: kõik algarvud, mis pole suuremad kui N.

Faili nimi: (V-NK-13) Eratosthenes.py

```
#================== Programm ============================
# Muutujad:
#     N -sisendfailist loetud arv N
#     sõel - sõnastiku tüüpi muutuja, mis sisaldab arvudele
#            vastavaid tüüpe (algarv/kordarv).
#            Näit: {...,5:"algarv",6:"kordarv",...}
#-------------------------------------------------------
from math import sqrt

fail = open("(V-NK-13).txt","r")
N = int(fail.readline())

fail.close()
sõel = {}
#täidab sõela
for i in range(2,N+1):
    sõel[str(i)] = "algarv"
#sõelumine, piisab sõeluda kuni ruutjuur(N)
for i in range(2,int(sqrt(N))+1):
    if sõel[str(i)] == "algarv":
        for j in range(i+i,N+1,i):
            sõel[str(j)] = "kordarv"

for i in range(2,N+1):
    if sõel[str(i)] == "algarv":
        print(i,end=" ")
#============= Programmi lõpp ==================
```

Näide programmi töö tulemsest:

Sisend: 171

Väljund: 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 107 109 113 127 131 137 139 149 151 157 163 167

5.1.11 Ülesanne V-NK-15 SÜT, VÜK

Antud: naturaalarv $n > 1$ ja naturaalarvud $a1, \dots, an$.

Leida: a) arvude $a1, \dots, an$ suurim ühistegur,

b) arvude $a1, \dots, an$ vähim ühiskordne.

Faili nimi: (V-NK-15) SÜT-VÜK.py

```
#=============== Programm ============================
def algtegurite_leidmine():
    arvude_algtegurid = {}
    for i in range(len(arvud)):
        algtegurid = [1]
        jagaja = 2
        jagatav = arvud[i]
        while jagatav > 1:
            if jagatav / jagaja == int(jagatav / jagaja):
                algtegurid.append(jagaja)
                jagatav /= jagaja
            else:
                jagaja += 1
        algtegurid.sort()
        arvude_algtegurid[arvud[i]] = algtegurid
    print("Algtegurid",arvude_algtegurid)
    return arvude_algtegurid
#---------- Sisendi lugemine ------------------
fail = open("(V-NK-15).txt","r")
mitu = (int(fail.readline()))
loearvud = fail.readline()
fail.close()
loearvud = loearvud.split(" ")
arvud = []
if mitu > len(loearvud):
    print("Sisendfailis ei ole piisavalt algandmeid")
    exit()
for i in range(mitu):
    arvud.append(int(loearvud[i]))
print(arvud)
#------------------
#algteguriteks jagamine
print("Leiame SÜT")
algtegurid = algtegurite_leidmine()
ühised = {}
ühised = algtegurid[arvud[0]]
```

```
for i in range(1, len(arvud)):
    vahetulemus = []
    for j in range(len(algtegurid[arvud[i]])):
        otsi = algtegurid[arvud[i]][j] # lihtsalt loetavuse huvides
        if otsi in ühised:
            vahetulemus.append(otsi)
            ühised.remove(otsi)
    ühised = vahetulemus
SÜT = 1
for i in range(len(ühised)):
    SÜT *= ühised[i]
print("Suurim ühistegur on ",SÜT)

#algteguriteks jagamine
print("Leiame VÜK")
algtegurid = algtegurite_leidmine()
vük_tegurid = []
for i in range(len(algtegurid[arvud[0]])):
    vük_tegurid.append(algtegurid[arvud[0]][i])
#ülejäänutest juurde
for i in range(1,len(arvud)):
    võrdle_tegureid = vük_tegurid
    for j in range(len(algtegurid[arvud[i]])):
        if algtegurid[arvud[i]][j] in võrdle_tegureid:
            võrdle_tegureid.remove(algtegurid[arvud[i]][j])
    vük_tegurid = vük_tegurid + algtegurid[arvud[i]]
    #print("vahe "+str(vük_tegurid))
vük = 1
for i in range(len(vük_tegurid)):
    vük = vük * vük_tegurid[i]
print("Vähim ühiskordne on",vük)
#================ Programmi lõpp ======================
```

Näide programmi töö tulemusest:

Sisend:

```
5
6 9 21 12 15
```

Väljund:

```
[6, 9, 21, 12, 15]
Leiame SÜT
Algtegurid {6: [1, 2, 3], 9: [1, 3, 3], 21: [1, 3, 7], 12: [1, 2, 2, 3],
15: [1, 3, 5]}
Suurim ühistegur on  3
```

```
Leiame VÜK
Algtegurid {6: [1, 2, 3], 9: [1, 3, 3], 21: [1, 3, 7], 12: [1, 2, 2, 3],
15: [1, 3, 5]}
Vähim ühiskordne on 1260
```

Selle ülesande lahendi kirjutamisel tekkis probleem sõnastik-tüüpi muutujaga *arvude_algtegurid*. Pythonis on sõnastik-tüüpi muutujatel huvitav omadus, et selle muutuja omistamisel teisele muutujale peab meeles pidama, et muutes uues muutujas sõnastiku liikmete väärtusi, muudetakse need ära ka algses sõnastikus. Seega oleks mõistlik kasutada mõnda teist tüüpi muutujat. Antud lahendis on lihtsalt pärast suurima ühisteguri leidmist vähima ühiskordse leidmiseks arvutatud uuesti kõikide arvude algtegurid.

5.1.12 Ülesanne V-G Punkt koos koordinaatidega

Geomeetria osas on soovituslikud eeltingimused: *Lõikude pikkusi mõõdame ülesannetes pikselites, nurki kraadides. Programm võiks joonistel tähistada punktid väikese täidetud ringiga, mille juurde on kirjutatud punkti tähis ja koordinaatidega punkti korral ka koordinaadid, näiteks •A(120,150). Sisendandmete kohta võib eeldada, et punktid paigutuvad üksteisest piisavalt kaugele ja tähised ei kata üksteist.*

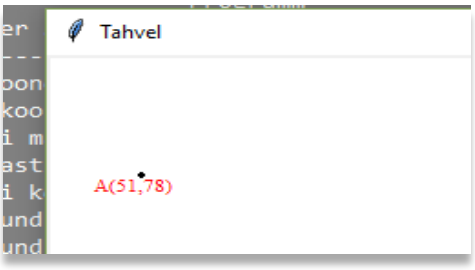
Faili nimi: (V-G) Punkt.py

```
#================ Programm =======================================
from tkinter import *
#----------------------------------------------------------------
# punkti joonestamiseks defineeritud funktsioon, mida saab kasutada hi-
lisemates lahendustes
def punkt(koordinaadid,**kwargs):
    #punkti märkimine joonisele,
    #kindlasti peab ette andma punkti koordinaadid. Näit. punkt((25,80))
    #punkti koordinaadid võivad arvutuste tulemusel olla mittetäisarvu-
lised. Ümardame need:
    x = round(koordinaadid[0],2)
    y = round(koordinaadid[1],2)
    #soovi korral võib anda ka punkti nime. Näit. punkt((25,80), nimi =
'A')
    nimi = kwargs.get('nimi',None)
    tahvel.create_oval(x-1,y-1,x+2,y+2,fill='black')
    #teksti pikkuse kontroll, et kiri tuleks keskele:
    if nimi != None:
        nimi=nimi+'('+str(x)+','+str(y)+')'
        kesk = int(len(nimi)/2)    #teksti keskkoht
        fondikorgus = 8
        tahvel.create_text(x-kesk,y+fondikorgus,font='Times
'+str(fondikorgus),text=nimi,fill='red')
    return
#----------------------------------------------------------------
raam = Tk()
raam.title("Tahvel")
tahvel = Canvas(raam, width=800, height=600, background="white")
fail = open("(V-G).txt","r")
avaldis = fail.readline()
fail.close()
punkti_tahis,x,y = avaldis.split(' ')
punkt((int(x),int(y)), nimi = punkti_tahis)
tahvel.pack()
raam.mainloop()
#================================================
```

Näide programmi töö tulemusest:

Sisend: `A 51 78`

Väljund: (joonisel 2)



Joonis 2 Ekraanitõmmis

5.1.13 Ülesanne V-G-1 Lõigu joonestamine punktide järgi

Antud: *kahe erineva punkti A ja B koordinaadid.*

Joonestada: *lõik AB*

Faili nimi: (V-G-1) Lõik kahe punktiga.py

```
#================== Programm =============================
from tkinter import *
#----------------------------------------------------------------
# Punkt koordinaatidega on kirjeldatud programmis (V-G) Punkt koordinaa-
tidega.py
def punkt(koordinaadid,**kwargs):
    x = round(koordinaadid[0],2)
    y = round(koordinaadid[1],2)
    nimi = kwargs.get('nimi',None)
    tahvel.create_oval(x-1,y-1,x+2,y+2,fill='black')
    if nimi != None:
        nimi=nimi+'('+str(x)+','+str(y)+')'
        kesk = int(len(nimi)/2)
        fondikorgus = 8
        tahvel.create_text(x-kesk,y+fondikorgus,font='Times
'+str(fondikorgus),text=nimi,fill='red')
    return
#------------------------------------------------------------
def lqik(A,B,**kwargs):  #Lõigu joonestamine punktide järgi
    # A ja B on lõigu otspunktide koordinaadid.
    # soovi korral võib lõigule ette anda lõigu värvi:
    vxrv = kwargs.get('vxrv', None)
    #    Lõigu    otspunktide    märkimiseks    kasutame    funktsiooni
punkt(koordinaadid,nimi)
    punkt(A, nimi = 'A')
    punkt(B, nimi = 'B')
    tahvel.create_line(A,B,fill=vxrv)
    return
#----------------------------------------------------------------------
raam = Tk()
raam.title("Tahvel")
tahvel = Canvas(raam, width=800, height=600, background="white")
#Algandmete lugemine
fail = open("(V-G-1).txt","r")
avaldis = fail.readline()
A = avaldis.split(',')
A[0] = int(A[0])
A[1] = int(A[1])
```

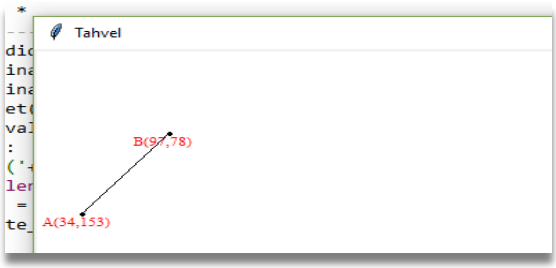
```
avaldis = fail.readline()
B = avaldis.split(',')
B[0] = int(B[0])
B[1] = int(B[1])
fail.close()

lqik(A,B)

tahvel.pack()
raam.mainloop()
#=============== Programmi lõpp ================================
```

Näide programmi töö tulemusest:

Sisend:

```
34,153
97,78
```

Väljund:



Joonis 3 Ekraanitõmmis

5.1.14 Ülesanne V-G-3 Sirge joonestamine punktide järgi

Antud: *kahe erineva punkti A ja B koordinaadid.*

Leida: *neid punkte läbiva sirge võrrand.*

Joonestada: *selline osa punkte A ja B läbivast sirgest, mis ulatub mõlemalt poolt ka väljapoole lõiku AB.*

Faili nimi: (V-G-3) Sirge kahe punktiga.py

```
#================ Programm ================================
from tkinter import *
#-------------- VAREM KIRJELDATUD FUNKTSIOONID ---------------
def punkt(koordinaadid,**kwargs):
    x = round(koordinaadid[0],2)
    y = round(koordinaadid[1],2)
    nimi = kwargs.get('nimi',None)
    tahvel.create_oval(x-1,y-1,x+2,y+2,fill='black')
    if nimi != None:
        nimi=nimi+'('+str(x)+','+str(y)+')'
        kesk = int(len(nimi)/2)
        fondikorgus = 8
        tahvel.create_text(x-kesk,y+fondikorgus,font='Times
'+str(fondikorgus),text=nimi,fill='red')
    return
#---------------- Sirge joonestamine ------------------------
def sirge(A,B,**kwargs):
    # Antud: kahe erineva punkti A ja B koordinaadid
    # Joonestada: selline osa punkte A ja B läbivast sirgest, mis ulatub
üle punktide A ja B
    #-----------------------------------------------------------
    # Joonestame punktid funktsiooniga punkt(koordinaadid,nimi)
    vxrv = kwargs.get('vxrv', None)
    punkt(A,nimi = 'A')
    punkt(B,nimi = 'B')
    #kahe punktiga määratud sirge võrrand (y-y1)(x2-x1)=(x-x1)(y2-y1)
[võrde põhiomadus]
    #eraldame koordinaadid
    x1,y1 = A
    x2,y2 = B
    #sirge võrrandi jaoks leiame sirge tõusu ja vabaliikme (y=ax+b)
    if x1 == x2:
        #kui x- koordinaadid kattuvad, on tegu vertikaalse sirgega ja
tõusu arvutamisel tuleks 0ga jagamine
        a = None
        b = x1
        print('x='+str(b))
```

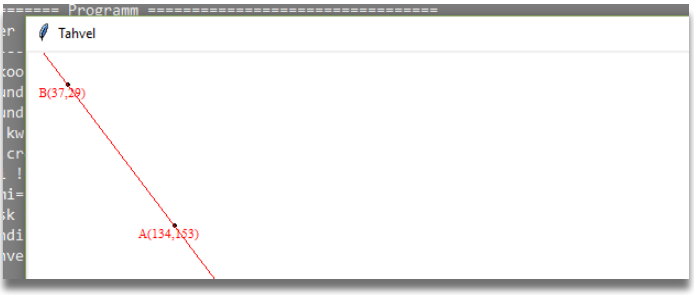
```
        tahvel.create_line((b,0),(b,600))
    else:
        a = (y2-y1)/(x2-x1)
        b = y1 - a * x1        #
        # kui tõus a = 0
        if a == 0:
            tahvel.create_line((0,b),(800,b),fill=vxrv)
        else:
            xa = int(-1 * (b / a))        # kui y = 0
            xl = int((600 - b) / a)       # kui y = tahvli alumine serv
            #print(xa,xl)
            tahvel.create_line((xa,0),(xl,600),fill=vxrv)
        # kuna tegemist on arvuti koordinaatteljestikuga, siis tuleb
sirge võrrandi väljakirjutamiseks võtta tõusu vastandarv
        print('y='+str(round(-1*a,2))+'x+'+str(round(b,2)))
    #kui peaks edaspidi vaja minema tõusu (a) ja vabaliiget (b), siis
anname need kaasa
    return a,b
#-------------------- sisend ----------------------------
raam = Tk()
raam.title("Tahvel")
tahvel = Canvas(raam, width=800, height=600, background="white")
fail = open("(V-G-3).txt","r")
avaldis = fail.readline()
A = avaldis.split(',')
A[0] = int(A[0])
A[1] = int(A[1])
avaldis = fail.readline()
B = avaldis.split(',')
B[0] = int(B[0])
B[1] = int(B[1])
fail.close()
# Kui on soov teist värvi sirget joonestada, siis on võimalik lisada
värv. Pole kohustuslik.
sirge(A,B,vxrv="red")
tahvel.pack()
raam.mainloop()
#==================== Programmi lõpp ==========================
```

Sisend:

134,153

37,29

Väljund: y=-1.28x+-18.3



Joonis 4 Ekraanitõmmis

5.1.15 Ülesanne V-G-5 Ringjoone joonestamine keskpunkti ja ringjoonel oleva punkti/raadiuse järgi

Antud: *ringi keskpunkti O koordinaadid ja*

a) ringjoonel asuva punkti A koordinaadid,

b) ringi raadius R.

Joonestada: *ringjoon, mille keskpunkt on O ja mis läbib punkti A/on raadiusega R.*

Faili nimi: (V-G-5) Ringjoon.py

```
#================== Programm ==================================
from tkinter import *
from math import *  # vajalik matemaatikapakett, et raadiuse arvutamise juures kasutada ruutjuurt
#-------------- VAREM KIRJELDATUD FUNKTSIOONID ------------------
def punkt(koordinaadid,**kwargs):
    x = round(koordinaadid[0],2)
    y = round(koordinaadid[1],2)
    nimi = kwargs.get('nimi',None)
    tahvel.create_oval(x-1,y-1,x+2,y+2,fill='black')
    if nimi != None:
        nimi=nimi+'('+str(x)+','+str(y)+')'
        kesk = int(len(nimi)/2)
        fondikorgus = 8
        tahvel.create_text(x-kesk,y+fondikorgus,font='Times '+str(fondikorgus),text=nimi,fill='red')
    return
#-------------------------------------------------------------
def ringjoon(O,R):
    #Antud: ringi keskpunkti O koordinaadid ja
    #a) ringjoonel asuva punkti A koordinaadid,
    #b) ringi raadius R.
    #Joonestada: ringjoon, mille keskpunkt on O ja mis läbib punkti A/on raadiusega R.
    #-------------------------------------------------------------
    # Joonestame keskpunkti funktsiooniga punkt(koordinaadid,nimi)
    x,y = O
    punkt((x,y),nimi = 'O')
    if type(R) == tuple:   # kui R on ringjoonel asuva punkti koordinaa-did
        x1 = int(R[0])
        y1 = int(R[1])
        x1 = x1 - x
        y1 = y1 - y
        R = sqrt(x1*x1+y1*y1)        # arvutame välja raadiuse
```

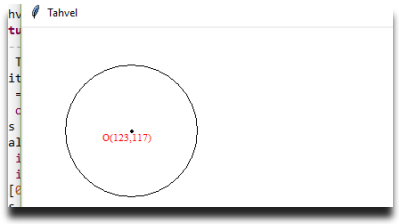
```
    tahvel.create_oval(x-R,y-R,x+R,y+R,outline="black")
    return
#--------------------------------------------------------------------
raam = Tk()
raam.title("Tahvel")
tahvel = Canvas(raam, width=800, height=600, background="white")
fail = open("(V-G-5).txt","r")
avaldis = fail.readline()
A = avaldis.split(',')
A[0] = int(A[0])
A[1] = int(A[1])
O = (A[0],A[1])
raadius = int(A[2])
fail.close()
ringjoon(O,raadius)
tahvel.pack()
raam.mainloop()
#================== Programmi lõpp ============================
```

Näited programmi töö tulemustest:

1) Sisend: Sisendfailis on ringi keskpunkti koordinaadid ja raadius

```
123,117,74
```

Väljund:

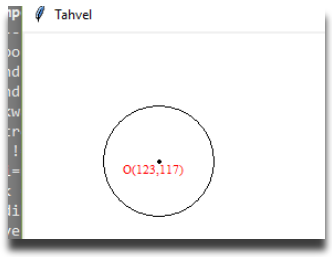


Joonis 5 Ekraanitõmmis

2) Sisend: Sisendfailis on ringi keskpunkti koordinaadid ja ringjoonel asuva punkti koordinaadid

123,117,50,20

Väljund:



Joonis 6 Ekraanitõmmis

Seda programmi võiks edasi täiendada, et joonisel oleks näha ka joonestatava ringi raadiuse väärtus.

5.1.16 Ülesanne V-G-9 Kahe ringjoone lõikepunktide/puutepunkti leidmine

Antud: *kahe ringjoone keskpunktid (x1,y1), (x2,y2) ja ringjoonte raadiused R1 ja R3.* ***Leida*** *ringjoonte lõikepunktide koordinaadid (kui lõikuvad)/puutepunkti koordinaadid (kui puutuvad)/negat vastus/otsus ühtimise kohta.*

Joonistada *ringjooned ning kirjutada lõikepunktide/puutepunkti juurde koordinaadid.*

Faili nimi: (V-G-9) Ringjoonte lõikumine.py

```
#=============== Programm ============================
from tkinter import *
import math
#-------------- VARASEMAST -----------------------------------
def punkt(koordinaadid,**kwargs):
    x = round(koordinaadid[0],2)
    y = round(koordinaadid[1],2)
    nimi = kwargs.get('nimi',None)
    tahvel.create_oval(x-1,y-1,x+2,y+2,fill='black')
    if nimi != None:
        nimi=nimi+'('+str(x)+','+str(y)+')'
        kesk = int(len(nimi)/2)
        fondikorgus = 8
        tahvel.create_text(x-kesk,y+fondikorgus,font='Times
'+str(fondikorgus),text=nimi,fill='red')
    return
#--------------------------------------------------------------
def lqik(A,B,**kwargs):
    vxrv = kwargs.get('vxrv', None)
    punkt(A, nimi = 'A')
    punkt(B, nimi = 'B')
    tahvel.create_line(A,B,fill=vxrv)
    return
#--------------------------------------------------------------
def ringjoon(O,R):
    x,y = O
    punkt((x,y),nimi = 'O')
    if type(R) == tuple:
        x1 = int(R[0])
        y1 = int(R[1])
        x1 = x1 - x
        y1 = y1 - y
        R = sqrt(x1*x1+y1*y1)
    tahvel.create_oval(x-R,y-R,x+R,y+R,outline="black")
    return
#--------------------------------------------------------------
```

```
def sirge(A,B,**kwargs):
    vxrv = kwargs.get('vxrv', None)
    punkt(A,nimi = 'A')
    punkt(B,nimi = 'B')
    x1,y1 = A
    x2,y2 = B
    if x1 == x2:
        a = None
        b = x1
        print('x='+str(b))
        tahvel.create_line((b,0),(b,600))
    else:
        a = (y2-y1)/(x2-x1)
        b = y1 - a * x1        #
        if a == 0:
            tahvel.create_line((0,b),(800,b),fill=vxrv)
        else:
            xa = int(-1 * (b / a))
            xl = (700 - b) / a
            tahvel.create_line((xa,0),(xl,600),fill=vxrv)
        print('y='+str(-1*a)+'x+'+str(b))
    return a,b
#-------------------------------------------------
def loikepunktid(KP1,r1,KP2,r2): # KP1 ühe ringi keskpunkti koordinaadid
                                 # KP2 teise ringi keskpunkti koordinaa-
did
    x1,y1 = KP1
    x2,y2 = KP2
    ringjoon(KP1,r1)
    ringjoon(KP2,r2)
    lqik(KP1,KP2,vxrv="blue")
    d = math.sqrt(math.pow(x2-x1,2)+math.pow(y2-y1,2)) # kahe ringjoone
keskpunktide kaugus
    # kontrolli, kas üldse on lõikepunktid võimalikud
    #1. ringjooned ei lõiku, sest vahekaugus on liiga suur
    if d > (r1 + r2):
        print("Antud ringjoontel puuduvad lõikepunktid, sest ringjooned
asuvad teineteisest liiga kaugel")
        return
    #3. ringjooned ühtivad
    if (d == 0) and (r1==r2):
        print("Antud ringjoontel puuduvad lõikepunktid, sest ringjooned
kattuvad")
        return
    #2. ringjooned ei lõiku, sest üks on teise sees
```

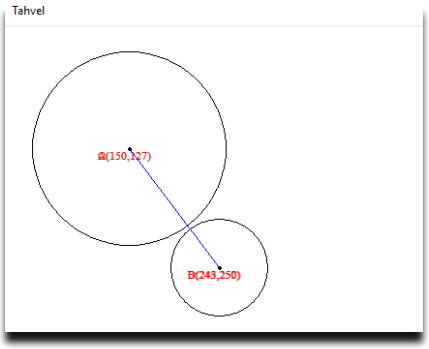
```
    if (max(r1,r2)-min(r1,r2)>d):
        print("Antud ringjoontel puuduvad lõikepunktid, sest üks ring-
joon asub teise sees")
        return

    a = (math.pow(r1,2)-math.pow(r2,2)+math.pow(d,2))/(2*d)
    x3 = x1 + a*(x2-x1)/d
    y3 = y1 + a*(y2-y1)/d
#--------
    h = math.sqrt(math.pow(r1,2)-math.pow(a,2))
    #sirge võrrand keskpunktidele
    if (x2 - x1) == 0:
        k = 0
    else:
        k=(y2-y1)/(x2-x1)
    b=(-1)*k*x1+y1
    u1=int(x3+h*(y2-y1)/d)
    v1=int(y3-h*(x2-x1)/d)
    u2=int(x3-h*(y2-y1)/d)
    v2=int(y3+h*(x2-x1)/d)
    punkt((u1,v1), nimi = 'K')
    punkt((u2,v2), nimi = 'K')
#------------
    return u1,v1,u2,v2
#------------------Sisendi lugemine--------------------------------
raam = Tk()
raam.title("Tahvel")
tahvel = Canvas(raam, width=800, height=600, background="white")
fail = open("(V-G-9).txt","r")
loerida = fail.readline()
esimene = loerida.split(',')
A = [int(esimene[0]),int(esimene[1])]
R1 = int(esimene[2])
loerida = fail.readline()
teine = loerida.split(',')
B = [int(teine[0]),int(teine[1])]
R2 = int(teine[2])
fail.close()

print(loikepunktid(A,R1,B,R2))

tahvel.pack()
raam.mainloop()
#========== Programmi lõpp ================================
```

Näiteid programmi töö tulemustest:

1) Sisend:

```
150,127,100
243,250,50
```

Väljund: `Antud ringjoontel puuduvad lõikepunktid, sest ringjooned asuvad teineteisest liiga kaugel`
`None`



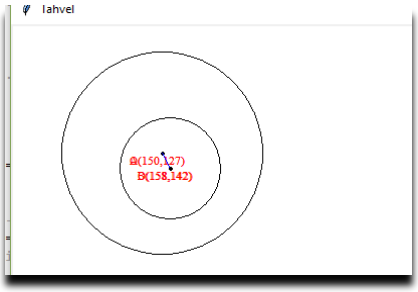
Joonis 7 Ekraanitõmmis

2) Sisend:

```
150,127,100
158,142,50
```

Väljund:

`Antud ringjoontel puuduvad lõikepunktid, sest üks ringjoon asub teise sees`
`None`

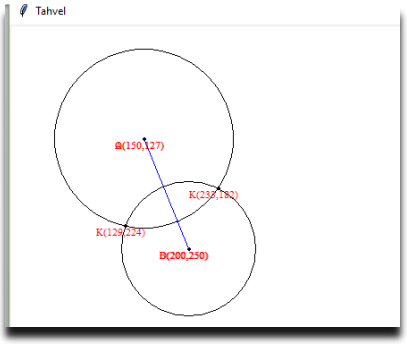


Joonis 8 Ekraanitõmmis

3) Sisend:

```
150,127,100
200,250,75
```

Väljund: (233, 182, 129, 224)

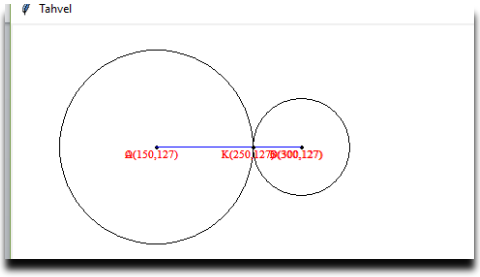


Joonis 9 Ekraanitõmmis

4) Sisend:

```
150,127,100
300,127,50
```

Väljund: (250, 127, 250, 127)



Joonis 10 Ekraanitõmmis

5.1.17 Ülesanne V-S-5 Mediaan

Antud: *klassi õpilaste arv n ($n \leq 40$) ja kõigi õpilaste pikkused.*

Leida: *õpilaste pikkuste mediaan.*

Faili nimi: (V-S-5) Mediaan.py

```
#================ Programm ================================
# Muutujad:
#    rida - sisendfailist esimeselt realt sisseloetud
#           õpilaste arv, järgnevad
#    õpilasi - õpilaste arv, täisarv
#    pikkused - pikkuste nimekiri, õpilaste arvu pikkune
#    keskkoht - poolitatud õpilaste arv
#    mediaan - pikkuste mediaan
fail = open("(V-S-5).txt","r")
rida = fail.readline()
õpilasi = int(rida)
pikkused = []
for i in range(õpilasi):
    rida = fail.readline()
    pikkused.append(int(rida))
fail.close()
print("Antud pikkused",pikkused)
#sorteerime õpilaste arvu jagu pikkusi,
#ül. tingimustest ülearu sisestatud pikkused jäävad
#arvestusest välja
järg = pikkused[0]
for i in range(1,õpilasi):
    if pikkused[i] < järg:
        välja = pikkused[i]
        pikkused.pop(i)
        j = 0
        while pikkused[j] < välja:
            j += 1
        pikkused.insert(j,välja)
    järg = pikkused[i]
print("Järjestatud pikkused",pikkused)
keskkoht = õpilasi/2
if keskkoht == int(keskkoht):
    mediaan = (pikkused[int(keskkoht)]+pikkused[int(keskkoht)+1])/2
else:
    mediaan = pikkused[int(õpilasi/2)]
```

```
print("Pikkuste mediaan on ",mediaan)
#============== Programmi lõpp ================================
```

Näide programmi töö tulemusest:

```
Antud pikkused [163, 165, 162, 123, 134, 145, 154, 163, 173, 178, 172,
173, 173]
Järjestatud pikkused [123, 134, 145, 154, 162, 163, 163, 165, 172, 173,
173, 173, 178]
Pikkuste mediaan on  163
```

5.1.18 Ülesanne VI-M-3 Laiendadamine antud nimetajani

Antud: murd kujul x/y ja nõutav uus nimetaja (positiivne naturaalarv a, mis on y kordne).
Leida: nimetajani a laiendatud murd.

Faili nimi: (VI-M-3) Laiendamine nimetajani.py

```
#================ Programm ============================
fail = open("(VI-M-3).txt","r")
loe_andmed = fail.readline()
fail.close()
jagatis, uus_nimetaja = loe_andmed.split(" ")
lugeja, nimetaja = jagatis.split("/")
lugeja = int(lugeja)
nimetaja = int(nimetaja)
antud_nimetaja = int(uus_nimetaja)
#-----------------------------------------------------
laiendaja = antud_nimetaja/nimetaja
if not laiendaja == int(antud_nimetaja/nimetaja):
    print("Uus nimetaja ei ole algse nimetaja kordne. Ei vasta ülesande tingimustele")
else:
    uus_lugeja = int(laiendaja * lugeja)
    uus_nimetaja = int(laiendaja * nimetaja)
    print("Murd "+str(lugeja)+"/"+str(nimetaja)+" laiendatud nimetajani "+str(antud_nimetaja)+" on ",end="")
    print(str(uus_lugeja)+"/"+str(uus_nimetaja))
#========== Programmi lõpp =====================
```

Näiteid programmi töö tulemustest:

1) Sisend: 7/8 104

Väljund: Murd 7/8 laiendatud nimetajani 104 on 91/104

2) **Sisend:** 2/7 99

Väljund: Uus nimetaja ei ole algse nimetaja kordne. Ei vasta ülesande tingimustele

Kokkuvõte

Bakalaureusetöö eesmärgiks oli luua ülesannete kogule “Programmeerime endale Lahendaja” näidislahendused, mis järgiks põhikooli matemaatika õpetamiseks kasutatavaid algoritme. Seega, võimalikult madala tasemega programmeerimiskeele funktsioonidega saada põhikooli matemaatika ülesannetele lahendused.

Töö jooksul sai uuritud Eestis programmeerimise õpetamisel kasutatavates ülesannete kogudes leiduvaid analoogseid ülesandeid. Samuti sai uuritud matemaatika õpikutes käsitletud ülesannete lahendamise algoritme.

Töö käigus valmisid näidislahendused ülesannete kogule. Lahendused on ideede allikaks programmeerimise õpetamisel ja õppimisel, kuidas oleks võimalik üht või teist probleemi lahendada.

Viidatud kirjandus

[1] Prank, R., Kaljuste T. Ülesannete kogu „Programmeerime endale Lahendaja“. http://didaktika-dev.cs.ut.ee/wp-content/uploads/2019/04/Programmeerime_matemaatika_algoritme_1504.pdf

(10.05.2019)

[2] Tennisberg, T., Gabrel, K. Võistlusprogrammeerimine. Tartu Ülikool. 2017. https://www.teaduskool.ut.ee/sites/default/files/teaduskool/oppetoo/voistlusprogrammeerimine_i_osa.pdf (07.05.2019)

[3] Tennisberg, T., Gabrel, K. Võistlusprogrammeerimine. II osa. Tartu Ülikool. 2018. https://www.teaduskool.ut.ee/sites/default/files/teaduskool/oppetoo/voistlusprogrammeerimine_ii.pdf (07.05.2019)

[4] Tõnso, T., Veelmaa, A. Matemaatika X klassile. 2004

[5] Levin, A., Tõnso, T., Veelmaa, A. Matemaatika XI klassile. 1995.

[6] Tõnso, T., Veelmaa, A. Matemaatika XII klassile. 1996.

[7] Eesti informaatikaolümpiaad. Eelvoor 06.12.2014. http://eio.ee/uploads/Main/2014-12-06-ev-pqh-et.pdf (05.05.2019

[8] Eesti informaatikaolümpiaad. Lõppvoor 14.02.2015. http://eio.ee/uploads/Main/2015-02-14-fv.zip (05.05.2019)

[9] Eesti informaatikaolümpiaad. Lahtine võistlus 12–18.10.2015. http://eio.ee/uploads/Main/2015-02-14-fv.zip (05.05.2019)

[10] Eesti informaatikaolümpiaad. Lahtine võistlus. Eelvoor 19.11.2016. http://eio.ee/uploads/Main/2016-11-19-ev-et.zip(05.05.2019)

[11] Eesti informaatikaolümpiaad Eelvoor 18.11.2017. http://eio.ee/uploads/Main/2017-11-18-ev-et.zip (05.05.2019)

[12] Eesti informaatikaolümpiaad Eelvoor 08.12.2018. http://eio.ee/uploads/Main/2018-12-08-ev.zip (05.05.2019)

[13] Eesti informaatikaolümpiaad. Harjutusvoor 10.11.2018 http://eio.ee/uploads/Main/2018-11-10-sortots-et.zip (05.05.2019)

[14] Balti informaatikaolümpiaad. 2001. https://oi.edu.pl/boi2001/index0055.html?id=11 (05.05.2019)

[15] Balti informaatikaolümpiaad. 2002. http://ims.mii.lt/olimp/senas/english/boi2002/index.html (05.05.2019)

[16] TÜ Arvutiteaduse instituudi programmeerimise algkursuse õpik“ http://progeopik.ut.ee (09.05.2019)

[17] Lints, A., Etverk, E. Matemaatika III klassile. 1986. Tallinn: Valgus.

[18] Nurk, E., Telgmaa, A. Matemaatika IV klassile. 1986. Tallinn: Valgus.

Lisad

Ülesannete kogu juurde kuuluvad näidislahendused on bakalaureusetööga kaasa pandud failis „Näidislahendused.zip“

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Tiiu Kaljuste (sünnikuupäev: 15.04.1965),

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose „Ülesannete kogu ' Programmeerime endale Lahendaja' näidislahendused", mille juhendaja on Rein Prank,

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

   1.2 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.
2. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Tiiu Kaljuste

13.05.2019