3 savaitė. Kartojimas. Veiksmai su masyvų elementais. Sumos, kiekio, vidurkio skaičiavimas

Užduočių rinkinys IV gimnazijos klasės mokiniams

1 lygis

1. **Drabužių parduotuvė.** Parduotuvėje prekiaujama vyriškais ir moteriškais drabužiais. Kiekvieną dieną yra vedama apskaita: užrašoma, kiek vienetų drabužių yra parduota ir pasižymima informacija apie parduotą vienetą – vyriškas ar moteriškas drabužis bei jo kaina. Parašykite programą, kuri rastų:
   - už kokią pinigų sumą parduota moteriškų ir vyriškų drabužių atskirai;
   - kiek vidutiniškai kainavo moteriškų ir vyriškų drabužių vienetas;
   - už kokią pinigų sumą parduota drabužių per dieną.

Pradinių duomenų failo **drabuziai_data.txt** pirmoje eilutėje nurodomas per dieną parduotų drabužių vienetų skaičius **n** ($1 \leq n \leq 100$); tolesnėse n eilučių įrašyta informacija apie kiekvieną parduotą vienetą: simbolis „v“ (vyriškas drabužis) arba „m“ (moteriškas drabužis) bei drabužio kaina eurais (realusis skaičius).
Skaičiavimų rezultatus pateikite dviejų skaičių po kablelio tikslumu faile **drabuziai_res.txt** kaip pateikta pavyzdyje. Jei kurio nors tipo drabužiais neprekiauta, reikia išvesti žodį NE.

Reikalavimai programai

- Duomenims saugoti naudokite masyvus.
- Pradinių duomenų skaitymo funkcija `void Skaityti()`.
- Funkcija `void Skaiciuoti()`, skaičiuojanti vyriškų arba moteriškų drabužių kainų sumą bei vidutinę drabužio kainą. Į funkciją reikia kreiptis 2 kartus: su vyriškų drabužių ir moteriškų drabužių kainų masyvais.
- Rezultatų rašymo į failą funkcija `void Rasyti()`.

drabuziai_data.txt	**drabuziai_res.txt**
2 m 245.70 m 150.99	MOTERISKI 396.69 198.35 VYRISKI NE 396.69

2. **Sodinukai.** Pranas augina medelių sodinukus, kuriuos po to parduoda kitiems sodininkams. Parduoti skirti sodinukai turi būti ne žemesni už **a1** centimetrų ir ne aukštesni už **a2** centimetrų. Prano medelyne auga **n** sodinukų. Rengdamasis rudeninei mugei Pranas išmatavo kiekvieno sodinuko aukštį ir surašė duomenis faile **sodinukai_data.txt**. Pirmoje eilutėje įrašytas sodinukų skaičius **n** ($1 \leq n \leq 100$). Tolesnėse **n** eilučių įrašyti sodinukų aukščiai (sveikieji skaičiai). Paskutinėje eilutėje – du sveikieji skaičiai **a1** ir **a2**, nurodantys, kokiame aukščių intervale, įskaitant intervalo galus, yra parduoti tinkamų sodinukų aukščiai. Jei mugei tinkamų sodinukų nėra, turi būti žodis NE.

Parašykite programą, kuri rezultatų faile **sodinukai_res.txt** išsaugotų:

- vidutinį visų medelyne augančių sodinukų aukštį vieno skaitmens po kablelio tikslumu;
- sodinukų, atrinktų rudeninei mugei, skaičių;
- atrinktų sodinukų vidutinį aukštį vieno skaitmens po kablelio tikslumu;
- pateiktų rudeninei mugei atrinktų sodinukų aukščių sąrašą vienoje eilutėje.

Reikalavimai programai

- Duomenims saugoti naudokite masyvus.
- Pradinių duomenų skaitymo funkcija `void Skaityti().`
- Funkcija `void Atrinkti()`, sudedanti į naują masyvą rudeninei mugei atrinktus sodinukus.
- Funkcija `double Vidurkis()`, grąžinanti apskaičiuotą vidutinį sodinuko aukštį. Į funkciją kreiptis 2 kartus: su visų sodinukų ir su atrinktų mugei sodinukų masyvu.
- Rezultatų rašymo į failą funkcija `void Rasyti().`

sodinukai_data.txt	**sodinukai_res.txt**
5 53 45 56 54 56 50 60	52.8 4 54.8 53 56 54 56

3. **Uždavinių reitingavimas. n** ($1 \leq n \leq 100$) mokinių sprendžia uždavinius iš uždavinių rinkinio, kuriame yra **u** ($1 \leq u \leq 10$) uždavinių. Kiekvienas mokinys, išsprendęs visus uždavinius, balsuoja už jam labiausiai patikusį uždavinį.

Pradinių duomenų failo **reitingavimas_data.txt** pirmoje eilutėje įrašytas uždavinius reitingavusių mokinių skaičius **n** ir rinkinio uždavinių skaičius **u**. Tolesnėse n eilučių įrašyta po vieną sveikąjį skaičių – kiekvieno mokinio pasirinkto uždavinio numeris.

Parašykite programą, kuri į rezultatų failą **reitingavimas_res.txt** išvestų kiekvieno uždavinio numerį ir jo surinktų balų skaičių.

Reikalavimai programai

- Duomenims saugoti naudokite masyvus.
- Pradinių duomenų skaitymo funkcija `void Skaityti().`
- Rezultatų rašymo į failą funkcija `void Rasyti().`

reitingavimas_data.txt	reitingavimas_res.txt
5 3 2 3 2 3 2	1 0 2 3 3 2

4. **Vėlavimas į pamoką.** Informatikos mokytoja labai pyksta, jei mokiniai vėluoja į pamokas. Ji toleruoja ne daugiau kaip **mv** minučių vėlavimą. Pamoka prasideda, kai laikrodis rodo **v** valandą, **m** minučių. Informatikos mobilioje grupėje mokosi **n** mokinių.

Mokytoja pradinių duomenų faile **velavimas_data.txt** surašė duomenis, kuriuos reikės apdoroti. Pirmoje pradinių duomenų failo eilutėje įrašyti du sveikieji skaičiai: keli mokiniai **n** mokosi mobilioje grupėje ir kelių minučių **mv** vėlavimą toleruoja mokytoja. Antroje eilutėje įrašyti 2 sveikieji skaičiai: pamokos pradžios valanda **v** ir minutė **m**. Tolesnėse **n** eilučių surašyta informacija apie kiekvieną mokinį: vardas ir mokinio atėjimo į klasę laikas (valanda ir minutė, sveikieji skaičiai).

Parenkite programą, kuri į rezultatų failą **velavimas_res.txt** surašytų rezultatus taip, kaip pateikta pavyzdyje (nurodomas mokinio vardas ir keliomis minutėmis anksčiau/vėliau mokinys atėjo į kabinetą. Jei mokinys atėjo anksčiau – teigiamas skaičius, jei vėliau – neigiamas, jei su skambučiu – nulis). Jei vėluojančių (arba nevėluojančių) mokinių nėra, tuomet rezultatų faile turi būti įrašyta „Veluojanciu nera“ („Neveluojanciu nera“).

velavimas_data.txt	velavimas_res.txt
7 5 12 15 Adomas 12 5 Erika 12 15 Dominykas 12 23 Paulius 12 18 Marius 11 57 Laurynas 12 0 Gytis 12 22	Nevelavo 5 mokiniai: Adomas 10 Erika 0 Paulius -3 Marius 18 Laurynas 15 Vidutiniskai vienas mokinys atejo anksciau, min: 8.0 Pavelavo 2 mokiniai: Dominykas -8 Gytis -7 Vidutiniskai vienas mokinys velavo, min: 7.5

Reikalavimai programai

- Naudokite masyvus.
- Pradinių duomenų skaitymo funkcija `void Skaitymas()`.
- Funkcija `int Minutes()`, skaičiuojanti, keliomis minutėmis per anksti (arba per vėlai) mokinys atėjo į kabinetą. Funkcija skaičiuoja **vieno mokinio** laiką ir grąžina apskaičiuotą reikšmę per funkcijos vardą.

- Funkcija `bool Tikrinimas()`, kuri nustato, ar mokinys neviršijo leistino vėlavimo laiko. Funkcija grąžina apskaičiuotą reikšmę per funkcijos vardą.
- Funkcija `double Vidurkis()`, skaičiuojanti vidutinį mokinio atėjimo anksčiau (vėlavimo) laiką. Funkcija grąžina apskaičiuotą reikšmę per funkcijos vardą. Turi būti 2 kreipiniai į funkciją: skaičiuojant vidutinį nevėluojančių ir vėluojančių laiką. Vidutinis atėjimo anksčiau (vėlavimo) laikas išvedamas vieno skaitmens po kablelio tikslumu.
- Rezultatų saugojimo faile funkcija `void Rasymas()`. Funkcijoje failas turi būti atidaromas papildymui. Į funkciją reikia kreiptis 2 kartus – su atėjusiais į pamoką laiku ir su vėluojančiais.

2 lygis

5. **Vasaros kelionės.** Kelionių agentūra „Novaturas“ siūlo platų kelionių pasirinkimą. Trys draugai rengiasi vasarą keliauti po Europą. Pirmasis draugas kelionei gali skirti **d1** eurų, antrasis – **d2**, trečiasis – **d3**. Reikia surasti, kurios kelionės tiks kiekvienam iš draugų ir kurios bus tinkamos visiems trims draugams.

Pradinių duomenų failo **keliones_data.txt** pirmoje eilutėje įrašyti 3 realieji skaičiai: pinigų suma, kurią kiekvienas draugas gali skirti kelionei. Antroje eilutėje įrašytas „Novaturo“ siūlomas kelionių po Europą skaičius **n**. Tolesnėse eilutėse nurodyta informacija apie kiekvieną kelionę: aplankomos šalys, kelionės trukmė, kelialapio kaina, kiek pinigų reikia turėti papildomoms išlaidoms.

Parenkite programą, kuri į rezultatų failą **keliones_res.txt** išvestų rezultatus taip, kaip pateikta pavyzdyje, nurodant šalies pavadinimą, kelionės trukmę ir bendras kelionės išlaidas. Jei kuriam nors draugui nėra tinkamos kelionės, tuomet rezultatų faile turi būti įrašyta: „Pirmam draugui tinkamu kelioniu nera“(„Antram draugui tinkamu kelioniu nera“, „Treciam draugui tinkamu kelioniu nera“, „Visiems draugams tinkamu kelioniu nera“).

keliones_data.txt	**keliones_res.txt**
700 650 1000 5 Italija 10 550 200 Prancuzija 12 620 150 Kroatija 8 440 120 Vokietija 10 560 170 Graikija 12 570 100	Pirmam draugui tinkamos keliones: Kroatija 8 560 Graikija 12 670 Vidutine keliones trukme dienomis yra: 10 Antram draugui tinkamos keliones: Kroatija 8 560 Vidutine keliones trukme dienomis yra: 8 Treciam draugui tinkamos keliones: Italija 10 750 Prancuzija 12 770 Kroatija 8 560 Vokietija 10 730 Graikija 12 670 Vidutine keliones trukme dienomis yra: 10 Visiems draugams tinkamos keliones: Kroatija 8 560 Vidutine keliones trukme dienomis yra: 8

Reikalavimai programai

- Naudokite masyvus.
- Pradinių duomenų skaitymo funkcija `void Skaitymas()`.
- Funkcija `int Suma()`, skaičiuojanti bendras vienos kelionės išlaidas. Funkcija grąžina apskaičiuotą reikšmę per funkcijos vardą.
- Funkcija `int Trukme()`, skaičiuojanti, kokia yra vieno keliautojo (bendrų kelionių) vidutinė trukmė. Funkcija grąžina apskaičiuotą reikšmę per funkcijos vardą. Skaičiuodami vidurkį naudokite sveikųjų skaičių dalybą. Į funkciją turi būti kreipiniai skaičiuojant kiekvieno draugo ir visiems draugams tinkančių kelionių vidutinę trukmę.
- Funkcija `bool Tikrinimas()`, kuri nustato, ar kelionė tinka visiems draugams. Funkcija grąžina apskaičiuotą reikšmę per funkcijos vardą.
- Rezultatų saugojimo faile funkcija `void Rasymas()`. Funkcijoje failas turi būti atidaromas papildymui. Į funkciją reikia kreiptis su kiekvieno draugo kelionių rinkiniu ir bendru kelionių rinkiniu.

6. **Miesto autobusai.** Panevėžio mieste yra **n** autobusų maršrutų. Skirtingų maršrutų autobusai aplanko skirtingą stotelių skaičių (tarp stotelių yra skirtingi atstumai), bei perveža skirtingus keleivių skaičius. Autobusų parko administracija atlieka tyrimą, kurio tikslas – sugrupuoti autobusus į 2 grupes. Pirmai grupei turi priklausyti tų maršrutų autobusai, kuriais pervežama ne mažiau kaip **k** keleivių ir maršruto ilgis didesnis kaip **m** metrų, antrai – visi likusieji.

Pradinių duomenų failo **autobusai_data.txt** pirmoje eilutėje įrašyti 3 sveikieji skaičiai: **n** – maršrutų skaičius, bei **k** ir **m** reikšmės. Tolesnėse **n** eilučių įrašyta informacija apie kiekvieną maršrutą – maršruto numeris (gali būti, pvz., 10A, rekomenduojama naudoti string tipą), pervežtų keleivių skaičius, stotelių skaičius ir atstumai tarp gretimų stotelių (sveikieji skaičiai, metrais): 1-2, 2-3, 3-4 ir t.t.

Parenkite programą, kuri į rezultatų failą **autobusai_res.txt** surašytų rezultatus taip, kaip pateikta pavyzdyje: nurodyti maršruto numerį, pervežtų keleivių skaičių ir bendrą maršruto ilgį. Jei kuriai nors grupei priskirtų maršrutų nėra, tuomet rezultatų faile turi būti įrašyta „Pirmos grupes marsrutu nera“ („Antros grupes marsrutu nera“).

autobusai_data.txt	**autobusai_res.txt**
5 20 2500 5 14 5 500 700 400 200 7 25 8 800 450 600 250 400 600 700 6A 12 7 450 220 400 750 560 420 10 42 9 450 520 600 320 290 560 470 300 10A 15 5 490 250 360 540	Pirmos grupes marsrutai: 7 25 3800 10 42 3510 Antros grupes marsrutai: 5 14 1800 6A 12 2800 10A 15 1640

Reikalavimai programai

- Naudokite masyvus.
- Pradinių duomenų skaitymo funkcija `void Skaitymas()`.
- Funkcija `int Ilgis()`, skaičiuojanti, koks yra **vieno** maršruto ilgis. Funkcija grąžina apskaičiuotą reikšmę per funkcijos vardą.

- Funkcija `bool Tikrinimas()`, kuri nustato, ar **vienas** maršrutas atitinka pirmos grupės maršrutams keliamus reikalavimus. Funkcija grąžina apskaičiuotą reikšmę per funkcijos vardą.
- Rezultatų saugojimo faile funkcija `void Rasymas()`. Funkcijoje failas turi būti atidaromas papildymui. Į funkciją reikia kreiptis 2 kartus – su pirmos ir su antros grupės maršrutais.

7. **Programavimo kontrolinio darbo pažymių pasiskirstymas.** Mokyklos dienyne mokytojas surašė programavimo kontrolinio darbo pažymius. Pažymiai kinta nuo 1 iki 10. Parašykite programą, kuri suskaičiuotų po kiek procentų yra 10-ukų, 9-ukų, ..., 1-ukų. Pradinių duomenų failo **kontrolinis_data.txt** pirmoje eilutėje įrašytas pažymių kiekis **n** ($1 \leq n \leq 20$), antroje eilutėje – kiekvieno mokinio programavimo kontrolinio darbo pažymiai vienas nuo kito atskirti tarpais.

Rezultatų faile **kontrolinis_res.txt** reikia įrašyti, kiek procentų kokių pažymių buvo parašyta už kontrolinį darbą: pažymys ir kiek procentų tokių pažymių parašyta dviejų skaitmenų po kablelio tikslumu.

Reikalavimai programai

- Naudokite sveikųjų skaičių masyvus.
- Pradinių duomenų skaitymo funkcija `void Skaityti()`.
- Funkcija `void Skaiciuoti()`, skaičiuojančią pažymių pasiskirstymą procentais.
- Rezultatų rašymo į failą funkcija `void Rasyti()`.

kontrolinis_data.txt	**kontrolinis_res.txt**
13 1 5 8 7 10 9 9 8 7 7 4 6 8	1 7.69 2 0.00 3 0.00 4 7.69 5 7.69 6 7.69 7 23.08 8 23.08 9 15.38 10 7.69

3 lygis

8. **Akmenukai (XV olimpiada, 2003).** Eilėje stovi n vaikų. Jie sunumeruoti nuo 1 iki n iš kairės į dešinę. Kiekvienas jų arba turi saujoje akmenukų, arba sauja yra tuščia. Vaikai paeiliui dalina akmenukus dešinėje jų stovintiems vaikams.

Pirmasis vaikas duoda vieną akmenuką antrajam vaikui, po to – vieną akmenuką trečiajam vaikui ir t. t. Taip jis dalindamas eina į dešinįjį eilės galą, po to atgal link savo vietos kiekvienam vaikui vėl duodamas po akmenuką. Dešiniajame eilės gale stovintis (n-asis) vaikas vis gauna po akmenuką, kai dalinantis akmenukus vaikas eina į priekį ir atgal.

Vaikas nustoja dalinti, kai pasibaigia akmenukai arba kai prieina savo vietą. Jeigu vaikui pasibaigia akmenukai dar nepriėjus savo vietos, jis tuščiomis grįžta į savo vietą. Jeigu vaikas prieina savo vietą turėdamas rankose akmenukų, jis stoja į savo vietą ir pasilieka sau likusius akmenukus. Tada analogiškai akmenukus dalina antrasis, trečiasis vaikai ir t. t. Paskutinysis (n-asis) vaikas, stovintis dešiniajame eilės gale, akmenukų nedalina.

Kadangi akmenukai dalinami tik dešinėje stovintiems vaikams, antrasis vaikas duos akmenukus tik trečiajam, ketvirtajam ir t. t., trečiasis – ketvirtajam, penktajam ir t. t. Parašykite programą, kuri suskaičiuotų, kiek akmenukų turės kiekvienas vaikas, kai baigsis dalinimas.

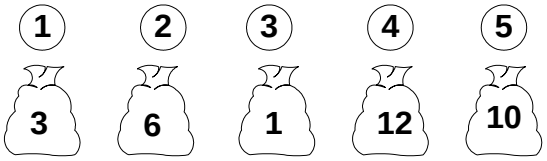
Pradiniai duomenys skaitomi iš failo **akmuo_data.txt**. Pirmoje eilutėje įrašytas vaikų skaičius ($3 \le n \le 10$). Antroje eilutėje įrašyta **n** tarpais atskirtų sveikųjų neneigiamų skaičių.

Rezultatai – (**n** skaičių) rašomi į pirmąją failo **akmuo_res.txt** eilutę.

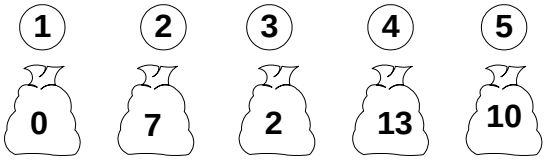
akmuo_data.txt	**akmuo_res.txt**
5 3 6 1 12 10	0 1 0 15 16

Pavyzdys, **n**=5 ir akmenukų skaičiai lygūs 3 6 1 12 10.

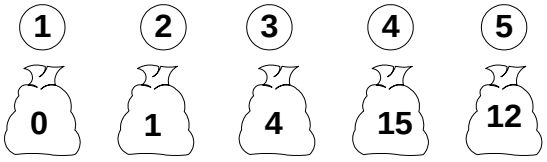
Prieš dalijimą:



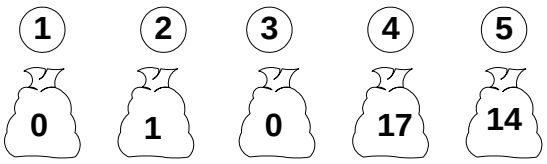
Pirmajam vaikui išdalinus akmenukus:



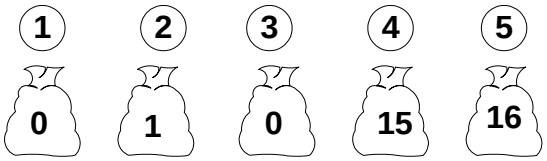
Antrajam vaikui išdalinus akmenukus:



Trečiajam vaikui išdalinus akmenukus:



Ketvirtajam vaikui išdalinus akmenukus:



Penktasis vaikas akmenukų nedalija. Baigus dalijimą turėtų būti išspausdinti tokie rezultatai:

```
0 1 0 15 16
```

9. **Konteineriai (XVIII olimpiada, 2006).** Transporto įmonė perveža krovinius konteineriais iš Vilniaus į Klaipėdą. Gamintojai prekes sukrauna į konteinerius, o į vieną konteinerį kraunami tik vieno gamintojo gaminiai. Taigi, gali būti ir nepilnų konteinerių. Pradėjus krauti konkretaus gamintojo prekes į konteinerį, kraunama tol, kol pasibaigia prekės arba konteineris pilnai prikraunamas.

Visų gamintojų gaminiai vienodo dydžio (t.y. vienodas gaminių kiekis telpa į konteinerį).
Parašykite programą, kuri apskaičiuotų kiek mažiausiai reikia konteinerių visiems gaminiams sutalpinti ir rastų kiek šie konteineriai bus užpildyti.
Pavyzdžiui, jeigu į konteinerį telpa 6 gaminiai, tai reikės rasti, kiek apskritai reikės konteinerių, keli konteineriai bus pilnai užpildyti (su 6 gaminiais), keli su penkiais gaminiais, keli su keturiais, trimis, dviem ir keli tik su vienu gaminiu.

Pradiniai duomenys įrašyti tekstiniame faile **konteineriai_data.txt**. Pirmoje eilutėje įrašyti du sveikieji skaičiai: gamintojų skaičius **N** ($1 \le N \le 1000$) ir konteinerio talpa **G** ($1 \le G \le 100$).
Tolesnėse **N** eilučių įrašyta kiek gaminių kiekvienas gamintojas nori pervežti į Klaipėdą. Tai sveikieji skaičiai nuo 1 iki 1000.
Rezultatai rašomi tekstiniame faile **konteineriai_res.txt**. Pirmoje eilutėje reikia įrašyti konteinerių skaičių, reikalingą visiems gaminiams pervežti.
Tolesnėse G eilučių spausdinti konteinerių skaičius pagal jų užpildymą mažėjimo tvarka. T. y. antroje eilutėje nurodoma kiek bus pilnai užpildytų konteinerių, trečioje – kiek bus konteinerių su G – 1 gaminiu, ketvirtoje – kiek bus konteinerių su G – 2 gaminiais ir t. t.

konteineriai_data.txt	**konteineriai_res.txt**	Paaiškinimai
1 5 26	6 5 0 0 0 1	Yra 1 gamintojas, o viename konteineryje telpa 5 gaminiai, reikia pervežti 26 prekes. Tam prireiks 6 konteinerių, iš jų 5 bus pilnai užpildyti (t. y. juose bus po 5 prekes) ir 1 konteineris bus pustuštis – jame bus tik 1 gaminys.

10. **Klaidos (XVII olimpiada, 2005).** Matematikos pamokoje mokytoja lentoje nubraižė stačiakampę lentelę, kurioje surašė įvairius sveikuosius skaičius. Užpildžius lentelę, mokytoja suskaičiavo kiekvienoje eilutėje surašytų skaičių sumą. Šias sumas ji įrašė į papildomą stulpelį lentelės pabaigoje. Tada mokytoja suskaičiavo kiekviename stulpelyje (įskaitant ir papildomą) surašytų skaičių sumas ir surašė jas į papildomą eilutę lentelės apačioje. Lentelėje paryškintas stulpelis ir eilutė, kuriuose surašytos sumos.

Pavyzdys

5	-2	4	3	**10**
1	-2	-5	2	**-4**
2	5	1	-4	**4**
-9	-7	1	6	**-9**
-1	**-6**	**1**	**7**	**1**

Milvydei lentelė patiko ir ji nutarė persirašyti ją į sąsiuvinį. Grįžusi namo mergaitė suprato, kad persirašydama lentelės duomenis ji buvo nepakankamai susikaupusi ir galėjo padaryti klaidą. Milvydė neabejoja, kad stulpelį ir eilutę, kuriuose surašytos sumos, ji persirašė teisingai. Taip pat ji visiškai įsitikinusi, kad galėjo padaryti ne daugiau kaip vieną klaidą.

Padėkite Milvydei išsiaiškinti, ar persirašydama lentelės duomenis ji padarė klaidą, ar ne. Jeigu klaida vis tik buvo padaryta, nurodykite, kurioje lentelės vietoje.

Pradiniai duomenys pateikiami faile **klaidos_data.txt**. Pirmoje eilutėje įrašyti du sveikieji skaičiai **N** ir **M** ($2 \le N, M \le 100$): N – lentelės eilučių skaičius, M – lentelės stulpelių skaičius. Kitose N eilučių pateikti lentelėje įrašyti skaičiai: kiekvienoje eilutėje įrašyta po M sveikųjų skaičių. Skaičiai lentelės viduryje patenka į intervalą [-100, 100], tuo tarpu stulpelių ir eilučių sumos gali būti didesnės už 100 arba mažesnės už -100.

Rezultatus įrašykite į failą **klaidos_res.txt**. Pirmoje eilutėje pateikite galimų klaidų skaičių – 0 arba 1. Jeigu klaida buvo padaryta, tai antroje eilutėje įrašykite lentelės eilutės ir stulpelio numerius, t. y. lentelės vietą, kurioje buvo padaryta klaida.

Pavyzdžiai

klaidos_data.txt	**klaidos_res.txt**	*Komentaras*
5 5 5 -2 4 3 10 1 -2 -5 2 -4 2 5 1 -4 4 -9 -7 1 6 -9 -1 -6 1 7 1	0	Klaidų nepadaryta
5 5 9 -2 4 3 10 1 -2 -5 2 -4 2 5 1 -4 4 -9 -7 1 6 -9 -1 -6 1 7 1	1 1 1	Vietoje 5 buvo įrašyta 9

11. **Juosta (XVII olimpiada, 2005).** Norgaudas sumanė pasigaminti žaidimą. Nupiešė kvadratėlių juostą, kvadratėlius sunumeravo nuo 1 iki N ir kiekviename jų parašė po kurį nors sveikąjį skaičių. Tada ant vieno iš kvadratėlio padėjo žaidimo figūrėlę ir sugalvojo tokias ėjimo taisykles:
    - Apskaičiuojamas skaičius *S* – tai dviejų skaičių sandauga. Pirmasis jų – tai skaičius, įrašytas kvadratėlyje, kuriame sustojo figūrėlė. Antrasis – tai skaičius, įrašytas kvadratėlyje, kuriama figūrėlė paskutinį kartą buvo sustojus. Žaidimo pradžioje antrasis skaičius lygus 1.
    - Jeigu skaičius *S* teigiamas, einama į dešinę per *S* kvadratėlių. Kai pasiekiamas paskutinis juostos kvadratėlis, tęsiama nuo pradžios – vėl nuo pirmojo juostos kvadratėlio.
    - Jeigu skaičius *S* neigiamas, einama į kairę per *S* kvadratėlių. Kai pasiekiamas pirmasis juostos kvadratėlis, tęsiama nuo pabaigos – nuo paskutinio juostos kvadratėlio.
    - Figūrėlei sustojus aprašytos taisyklės taikomos iš naujo.

Žaidimo tikslas – eiti tol, kol figūrėlei pavyks sustoti pageidaujamame kvadratėlyje. Padėkite Norgaudui sužinoti, kiek kartų figūrėlė sustos, kol pavyks sustoti nurodytame kvadratėlyje, žinoma, jei tai iš viso įmanoma.

Pradinis sustojimas (ant pradinio langelio) neskaičiuojamas, o paskutinysis (ant galinio langelio) – skaičiuojamas.

Pradiniai duomenys pateikti faile **juosta_data.txt**, kurį sudaro trys eilutės. Pirmoje eilutėje įrašytas vienas skaičius **N** ($2 \le N \le 100$) – kvadratėlių skaičius juostoje.

Antroje eilutėje įrašyta **N** tarpais atskirtų sveikųjų skaičių – juostos kvadratėliuose įrašyti skaičiai iš intervalo [-100; 100]. Trečioje eilutėje įrašyti du skirtingi skaičiai iš intervalo [1; N]: figūrėlės pradžios kvadratėlio numeris ir kvadratėlio, kuriame figūrėlė turi sustoti, numeris.

Rezultatų faile **juosta_res.txt** įrašykite, kiek kartų figūrėlė sustos, kol galiausiai atsidurs (sustos) norimame kvadratėlyje. Jei figūrėlė negali sustoti nurodytame kvadratėlyje, tai rezultatų faile įrašykite skaičių 0.

juosta_data.txt	**juosta_res.txt**
6 -1 2 3 -3 5 2 4 6	0
10 -5 3 -9 4 6 -4 1 -1 5 -7 2 1	8