1. Arvutipood (algajad)

Arvatavasti lihtsaim võimalus selle ülesande lahendamiseks on lähtuda tähelepanekust, et tegelikult on ainult 10 võimalikku päringut. Seega piisab, kui loeme faili viimaselt realt kõigepealt eraldi 4 sümbolit, mis määravad päringu liigi ja siis täisarvu, mis määrab uuritava parameetri piirväärtuse. Edasi võime iga päringuliigi jaoks kirjutada eraldi tingimuslause. Selline lahendus on toodud failis `apoodlah1.c`.

Teine võimalus on kirjutada tingimuse kontroll natuke üldisemalt. Failis `apoodlah2.c` toodud lahendusse tuleks rohkemate parameetrite (näiteks ka vahemälu suuruse, võrgu- ja helikaardi olemasolu jne) töötlemise võimaldamiseks teha muutusi ainult kahel real.

Testid

1. $N = 0$. Tingimus $H \geqslant 1000$, aga see tegelikult ei loe. Vastus EI OLE. 3 punkti.
2. $N = 2$. Tingimus $H \leqslant 343$, üks arvuti on täpselt piiripealse väärtusega. 3 punkti.
3. $N = 2$. Tingimus $P \leqslant 1200$, üks arvuti on täpselt piiripealse väärtusega. 3 punkti.
4. $N = 5$. Tingimus $P \geqslant 1000$. 3 punkti.
5. $N = 5$. Tingimus $M \leqslant 100$. 3 punkti.
6. $N = 5$. Tingimus $M \geqslant 256$. 3 punkti.
7. $N = 15$. Tingimus $V \leqslant 1000$. 3 punkti.
8. $N = 100$. Tingimus $V \geqslant 5000$. 3 punkti.
9. $N = 1000$. Tingimus $K \leqslant 900$. 3 punkti.
10. $N = 10\,000$. Tingimus $K \geqslant 9400$, üks arvuti on täpselt piiripealse väärtusega. 3 punkti.

Kokku 30 punkti.

2. Arvutipood (edasijõudnud)

Selle ülesande lahendamiseks on ilmselt vaja tingimus JA-sõnade kohalt osadeks tükeldada.

Esimene võimalus on kõik tingimused mingisse loendisse koguda ja käia selle loendiga läbi kõik arvutid, kontrollides igaühe juures, kas ta rahuldab kõiki tingimusi. Selline lahendus on toodud failis `epoodlah1.c`.

Teine võimalus on vaadelda saadud tingimusi ükshaaval, käies iga tingimusega läbi kõik arvutid ja heita kõrvale need, mis mõnda tingimust ei rahulda. Pärast kõigi tingimuste vaatlemist alles jäänud arvutid rahuldavad neid kõiki ja ongi otsitav vastus. Selline lahendus on toodud failis `apoodlah2.c`.

Testid

1. $N = 0$. Vastus EI OLE. 3 punkti.
2. $N = 2$. Üheosaline tingimus $H \geqslant 2323$, üks arvuti on täpselt piiripealse väärtusega. 3 punkti.
3. $N = 2$. Üheosaline ingimus $P \leqslant 1200$, üks arvuti on täpselt piiripealse väärtusega. 3 punkti.
4. $N = 5$. Mitmeosaline tingimus, mis sisaldab ühele muutujale kaht piirangut: $P \geqslant 100$ ja $P \geqslant 0$ (selles järjekorras, st esimene piirang on rangem). 3 punkti.
5. $N = 5$. Mitmeosaline tingimus, aga igaühest eraldi piisaks. Vastus EI OLE. 3 punkti.
6. $N = 5$. Mitmeosaline tingimus, mis sisaldab ühele muutujale kaht piirangut: $K \leqslant 80$ ja $K \leqslant 60$ (selles järjekorras, st teine piirang on rangem). 3 punkti.
7. $N = 15$. Mitmeosaline tingimus, piirab iga parameetrit vahemikuga. 3 punkti.
8. $N = 100$. Mitmeosaline tingimus, piirab iga parameetrit ühest suunast. 3 punkti.
9. $N = 1000$. Mitmeosaline tingimus, piirab iga parameetrit vahemikuga. 3 punkti.
10. $N = 10\,000$. Mitmeosaline tingimus, piirab iga parameetrit vahemikuga. 3 punkti.

Kokku 30 punkti.

3. Lukud

Selle ülesande lahenduse põhialgoritm peaks olema ilmne: lugeda algul sisse koopa kaart ja seejärel ükshaaval liikumiskäsud, neid jooksvalt täites. Kui mingi käsu täitmisel tekib viga, väljastada EI ja selle number; kui satume võtme juurde, märkida, et see võti on nüüd käes; kui satume väljapääsu juurde ja meil juba kõik võtmed käes, väljastada JAH ja käsu number. Kui kõik käsud on lõpuni täidetud ja midagi muud pole väljastatud, väljastada EI.

Ülesande raskus (koopas liikumise tehniliselt korrektse realiseerimise kõrval) on selles, et õigesti arvet pidada juba leitud võtmete üle. Ilmselt ei piisa ainult võtmeruutude läbimiskordade loendamisest, sest aardeotsija võib läbida mõne võtme asukohat korduvalt ja mõne teise võtme asukohta üldse mitte.

Üks võimalus oleks pidada eraldi loendit kõigist võtmetest ja märkida võtme leidmisel selle olek. Siis peab iga kord väljapääsu juurde sattudes kontrollima kõigi võtmete olekuid.

Parem lahendus on simuleerida võtme leidmist nii nagu see reaalselt toimuks: kui aardeotsija esimest korda võtme juurde satub, võtab ta selle üles ja edaspidi on selle koha peal tavaline käik. Siis ei ole vaja karta, et me üht võtit mitmekordselt loeks ja seega võime väljapääsu juurde sattudes lihtsalt võrrelda juba leitud võtmete arvu üldse kaardil olnud võtmete arvuga. Selline lahendus on toodud failis `lukudlah1.pas`.

Testid

1. $N = 3$, $M = 3$, $L = 3$. Võtmeid pole, liikuda tuleb ainult otse edasi, väljumine enne käskude lõppu.
   Vastus: JAH, 2. 2 punkti.
2. $N = 3$, $M = 3$, $L = 16$. 1 võti, pöörded ainult vasakule, väljumine viimasel käsul.
   Vastus: JAH, 16. 2 punkti.
3. $N = 4$, $M = 4$, $L = 58$. 2 võtit, pöörded ainult paremale, väljumine viimasel käsul.
   Vastus: JAH, 58. 2 punkti.
4. $N = 5$, $M = 5$, $L = 24$. Hulk võtmeid, pöörded mõlemas suunas, väljumine viimasel käsul.
   Vastus: JAH, 24. 2 punkti.
5. $N = 20$, $M = 20$, $L = 80$. Suur hõre juhuslik test.
   Vastus: JAH, 79. 2 punkti.
6. $N = 20$, $M = 20$, $L = 237$. Suur spiraalikujuline labürint.
   Vastus: JAH, 237. 2 punkti.
7. $N = 3$, $M = 3$, $L = 1$. Jookseb vastu seina.
   Vastus: VIGA, 1. 2 punkti.
8. $N = 3$, $M = 3$, $L = 4$. Jookseb kaardilt välja lääne suunas.
   Vastus: VIGA, 4. 2 punkti.
9. $N = 3$, $M = 3$, $L = 4$. Jookseb kaardilt välja ida suunas.
   Vastus: VIGA, 3. 2 punkti.
10. $N = 3$, $M = 3$, $L = 9$. Jookseb kaardilt välja lõuna suunas.
    Vastus: VIGA, 6. 2 punkti.
11. $N = 3$, $M = 3$, $L = 4$. Jookseb kaardilt välja põhja suunas.
    Vastus: VIGA, 4. 2 punkti.
12. $N = 3$, $M = 3$, $L = 1$. Jõuab väljapääsu juurde, kuid võtmed kogumata.
    Vastus: EI. 2 punkti.
13. $N = 3$, $M = 3$, $L = 1$. Kogub võtmed, kuid ei jõua väljapääsu juurde.
    Vastus: EI. 2 punkti.
14. $N = 3$, $M = 3$, $L = 3$. Käib väljapääsu juures, kuid kogub võtmed alles hiljem.
    Vastus: EI. 2 punkti.
15. $N = 3$, $M = 3$, $L = 5$. 2 võtit, käib ühe juures 2 korda, siis väljapääsu juurde.
    Vastus: EI. 2 punkti.

Kokku 30 punkti.

4. Sierpinski klots

Kuna iga kõrgemat järku vaip on saadud madalamat järku vaibast tükkide väljalõikamisega, piisab ülesande lahendamiseks, kui leiame kõige madalamat järku vaiba, millel on vastaval kohal auk. Kui see järk on X, siis on otsitav vastus $N - X + 1$.

Arvutamist alustame kõige kõrgemat järku vaibast ja liigume vaiphaaval allapoole. Kontrollimaks, kas vaadeldaval vaibal on antud koordinaatide kohal auk, on mugavam kasutada 0-st algavaid koordinaate. Siis kehtib reegel: vaibas on kohal (r, v) auk parajasti siis, kui nii r kui ka v annavad 3-ga jagades jäägi 1 või mõnel selle vaiba all oleval vaibal on samas kohas auk. Et teisendada (r, v) alumise vaiba jaoks, peame nii r kui ka v jagama (täisarvuliselt) 3-ga. Nii saame ülemisest vaibast alustades alumiseni liikudes ja jooksvalt koordinaate teisendades leida kõige alumise vaiba, millel on etteantud koordinaatidel auk. Ühegi vaiba täiskaarti konstrueerida polegi vaja.

Sellel ideel põhinev lahendus on toodud failis `klotslah1.pas`.

Testid

1. $N = 1$, $R = 1$, $V = 1$. Vastus 0. 3 punkti.
2. $N = 1$, $R = 2$, $V = 2$. Vastus 1. 3 punkti.
3. $N = 3$, $R = 15$, $V = 24$. Vastus 2. 3 punkti.
4. $N = 4$, $R = 41$, $V = 39$. Vastus 4. 3 punkti.
5. $N = 10$, $R = 22222$, $V = 23456$. Vastus 10. Siit alates enam tervet tabelit konstrueerida ei jõua. 3 punkti.
6. $N = 11$, $R = 42149$, $V = 32834$. Vastus 4. 3 punkti.
7. $N = 12$, $R = 145837$, $V = 393751$. Vastus 3. 3 punkti.
8. $N = 13$, $R = 497191$, $V = 1480684$. Vastus 2. 3 punkti.
9. $N = 14$, $R = 2445167$, $V = 1161551$. Vastus 1. 3 punkti.
10. $N = 15$, $R = 3484647$, $V = 7335495$. Vastus 0. 3 punkti.

Kokku 30 punkti.

5. Summad

Selle ülesande lahendamiseks pole üldiselt muud võimalust kui kõigi variantide läbivaatus: proovime paigutada igasse kahe arvu vahekohta nii pluss- kui miinusmärki ja arvutame iga võimaluse jaoks avaldise väärtuse.

Klassikaline viis kõigi variantide läbivaatuse programmeerimiseks on tagurdusmeetud: kui meil on juba leitud mingi kombinatsioon esimese k märgi paigutamiseks, siis on meil järgmiste märkide paigutamiseks kõige lihtsam lahendada rekursiivselt sama ülesannet.

Kui kõigi märkide paigutamise järel on avaldise väärtuseks otsitav arv, väljastame tulemuse ja lõpetame kohe otsingu. Selline lahendus on toodud failis `summalah0.pas`.

Kui meil on jadas N arvu, siis on nende vahele märkide paigutamiseks kokku 2^{N-1} võimalust ($N-1$ positsiooni, ja igaühes võib teistest sõltumatult olla üks kahest võimalikust märgist). Seega on halvimal juhul vaja läbi vaadata $2^2 3 = 8\,388\,608$ võimalikku märkide paigutust. Kui me seejuures teeme iga võimaliku paigutuse juures summa väljaarvutamiseks N liitmis- või lahutamistehet, annab see halvimal juhul tulemuseks umbes 200 miljonit tehet, mida testimismasin ühe sekundiga ära teha ei jõua.

Hea võimalus programmi kiirendada juba paigutatud märkide põhjal avaldise algusosa väärtus jooksvalt välja arvutada ja see rekursiivsele otsingule kaasa anda. Siis peame iga lõpliku variandi juures lisama summasse ainult ühe arvu ja saame $N-1$ "liidetavaga" osasummat taaskasutada 2 korda, $N-2$ "liidetavaga" osasummat 4 korda jne. Selline lahendus, mis on toodud failis `summalah1.pas`, mahub kõigis testides vabalt ajapiiridesse.

Testid

1. $N = 3$. Lahend on ühene. 5 punkti.
2. $N = 3$. Lahend on ühene. 5 punkti.
3. $N = 3$. Lahend puudub. 5 punkti.
4. $N = 4$. Lahend pole ühene. 5 punkti.
5. $N = 24$. Lahend on ühene. 5 punkti.
6. $N = 24$. Lahend on ühene, vastuses kõik plussmärgid (ebasoodsaim variant neile, kes esimesena proovivad igasse positsiooni miinusmärki). 5 punkti.
7. $N = 24$. Lahend on ühene, vastuses kõik miinusmärgid (ebasoodsaim variant neile, kes esimesena proovivad igasse positsiooni plussmärki). 5 punkti.
8. $N = 24$. Lahend puudub. 5 punkti.

Kokku 40 punkti.

6. Sulgavaldised

Üsna lihtne võimalus selle ülesande lahendamiseks on genereerida kõik N sulust koosnevad jadad, aga väljastada ainult need, mis osutuvad korrektseteks avaldisteks. Avaldise korrektsuse kontrollimiseks võime alustada tähelepanekust, et korrektses avaldises leidub alati vähemalt üks "kõige sisemine" alamavaldis kujul `()`, `[]` või {} ja selle eemaldamisel peab tulemuseks olema jälle korrektne avaldis. Seega peame korrektest avaldisest sulupaare eemaldades saama lõpuks tühja sõne. Sellel ideel põhinebki failis `suludlah0.pas` toodud lahendus, mis aga on suuremates testides liiga aeglane, sest teeb palju tühja tööd mittekorrektsete avaldiste genereerimisel.

Parema lahenduse saame, kui genereerime sulujadasid nii, et saamegi kohe ainult korrektsed avaldised ja midagi lisaks kontrollida pole enam vaja. Selleks tasub kõigepealt uurida, millistest osadest üks korrektne avaldis üldse koosneda võib. Üks tähelepanek on, et pikemaid avaldisi saab lühematest koostada kahel põhimõtteliselt erineval viisil:

1. kui A on korrektne n-märgiline avaldis, siis `(`A`)`, `[`A`]` ja $\{A\}$ on korrektsed $n+2$-märgilised avaldised (neid nimetame lihtavaldisteks);
2. kui A on korrektne n-märgiline ja B on korrektne m-märgiline avaldis, siis AB on korrektne $n+m$-märgiline avaldis (neid nimetame liitavaldisteks).

Kui loeme ka tühja sõne korrektseks avaldiseks, siis on kõik pikemad avaldised võimalik saada tühjadest sõnedest kahe eelmise reegli korduva rakendamisega. Muidugi peab nende reeglite järgi avaldiste genereerimisel olema ettevaatlik, et vältida sama avaldise korduvat genereerimist. Näiteks avaldise `()[]()` saame konstrueerida nii kujul `()[]+()` kui ka kujul `()+[]()`. Korduva genereerimise vältimiseks võime nõuda, et reegli 2 rakendamisel peab B alati olema reegli 1 järgi saadud avaldis. Sellel ideel põhinebki failis `suludlah1.pas` toodud lahendus, mis töötab küll kiiresti, aga vajab üsna palju mälu, sest hoiab töö ajal kõiki avaldisi korraga mälus.

Muide, kui vaja on ainult avaldiste arvu, piisab vaid $2N$ täisarvu mälus hoidmisest. Sellel ideel põhineb välkkiirelt töötav avaldiste arvu leidmise lõik failis `suludtester.cpp` olevas vastuste kontrollijas.

Veel parema lahenduse saame, kui lähtume tähelepanekust, et korrektses sulgavaldises vastab igale algavale sulule täpselt üks lõppev sulg ja mistahes algavatele sulgudele vastavad lõppevad sulud on avaldises algavatega võrreldes peegelpildis. See tähendab, et kui me oleme juba genereerinud mingi sulgude jada S, mille avatud, aga veel sulgemata sulgude "paariliste" jada on T, siis on meil jada S pikendamiseks järgmised võimalused:

1. lisame S lõppu algava sulu; siis peame lisama T algusesse ka vastava lõppeva sulu;
2. sulgeme jada S parempoolseima veel sulgemata sulu; siis peame vastava sulgeva sulu T algusest eemaldama. (Oluline on panna tähele, et nende kahe reegli järgimine tagab, et jada T esimene sulg on alati jada S parempoolseima veel sulgemata sulu paariline.)

Sellel ideel põhinev lahendus on toodud failis `suludlah2.pas`, mis töötab kiiresti ja vajab mälu ainult ühele avaldisele: sellele, mida ta parajasti genereerib.

Testid

1. $M = 1$, $N = 8$. 14 avaldist. 4 punkti.
2. $M = 1$, $N = 12$. 132 avaldist. 4 punkti.
3. $M = 1$, $N = 16$. 1430 avaldist. 4 punkti.
4. $M = 1$, $N = 20$. 16 796 avaldist, naiivne lahendus jääb liiga aeglaseks. 4 punkti.
5. $M = 2$, $N = 8$. 224 avaldist. 4 punkti.
6. $M = 2$, $N = 10$. 1344 avaldist, naiivne lahendus ajapiiri peal. 4 punkti.
7. $M = 2$, $N = 12$. 8448 avaldist, naiivne lahendus jääb liiga aeglaseks. 4 punkti.
8. $M = 3$, $N = 6$. 135 avaldist. 4 punkti.
9. $M = 3$, $N = 8$. 1134 avaldist, naiivne lahendus ajapiiri peal. 4 punkti.
10. $M = 3$, $N = 10$. 10 206 avaldist, naiivne lahendus jääb liiga aeglaseks. 4 punkti.

Kokku 40 punkti.