A picture containing shape

Description automatically generated

SEMINARSKA NALOGA

PROGRAMIRANJE 1

PRAKTIČNA MATEMATIKA

Avtor: Luka Zagorc

Ljubljana 2022/23

|  |  |  |  |
| --- | --- | --- | --- |
|  | Ime naloge | Aktivna povezava | Število težavnostnih točk |
| 1.del |  |  |  |
|  | Moscow Dream | <https://open.kattis.com/problems/moscowdream> | 2.6 T |
|  | Air Conditioned Minnions | <https://open.kattis.com/problems/airconditioned> | 3.3 T |
|  | Baby Bites | <https://open.kattis.com/problems/babybites> | 1.9 T |
|  | Boss Battle | <https://open.kattis.com/problems/bossbattle> | 1.8 T |
|  | Simon Says | <https://open.kattis.com/problems/simon> | 3.0 T |
|  | Millionaire Madness | <https://open.kattis.com/problems/millionairemadness> | 2.6 T |
|  |  |  |  |
|  | Skupaj točk |  | 15.2 T |
|  |  |  |  |
| 2.del |  |  |  |
|  | Guessing Game | <https://open.kattis.com/problems/guessinggame> | 3.1 T |
|  | Knight Jump | <https://open.kattis.com/problems/knightjump> | 2.4 T |
|  | Egypt | <https://open.kattis.com/problems/egypt> | 2.1 T |
|  | Collatz Conjecture | <https://open.kattis.com/problems/collatz> | 4.2 T |
|  | Rational Arithmetic | <https://open.kattis.com/problems/rationalarithmetic> | 3.5 T |
|  |  |  |  |
|  | Skupaj točk |  | 15.3 T |
|  |  |  |  |

**IZJAVA**

Luka Zagorc izjavljam, da sem seminarsko nalogo opravil samostojno in da sem njen avtor. To pomeni, da sem v poročilu točno označil vse tiste dele kode, ki sem jih povzel iz virov in vire navedel. Prav tako vem, da izjava o avtorstvu pomeni, da znam razložiti vsako podrobnost kode, ko jo oddajam. Zavedam se, da v primeru, če izjava prvega stavka ni resnična, kršim disciplinska pravila.

**POROČILO 1. DEL**



MOSCOW DREAM



Naloga je na spletni strani [https://open.kattis.com/problems/moscowdream](https://open.kattis.com/problems/moscowdream%20in%20je%20vredna%202.6) in je vredna 2.6 točk.



**BESEDILO NALOGE**

Za veliko študentov je velik dosežek dobiti možnost udeležbe na ICPC World Finals. ECPC Asia Danang Regional Contest 2019 je priložnost, kjer lahko uresničijo svoje sanje. Za druge pa priložnost, da si malo napnejo možgane in rešujejo zanimive probleme.

Mi, v znanstveni komisiji to razumemo in smo poskusili sestaviti nabor nalog, ki bodo zanimive in raznovrstne tako glede teme kot tudi težavnosti. Nekaj mesecev smo sprejemali predloge od številnih ljudi. Naloge so razvrščene po težavnosti: a lahke, b srednje težke in c težke. Na podlagi predlogov, bi radi oblikovali nabor nalog, ki so:

* sestavljene natančno iz n nalog,
* vsebujejo vsaj 1 lahko nalogo,
* vsebujejo vsaj 1 srednje težko nalogo,
* vsebujejo vsaj 1 težko nalogo.

Tvoja naloga je ugotoviti, če je možno sestaviti nabor nalog, tako da so zgornji pogoji izpolnjeni.

**Vhodni podatki**

Vhodni podatki vsebujejo 4 cela števila a, b, c in n (0 ≤ a, b, c ≤ 10,1 ≤ n ≤ 20).

**Izhodni podatki**

Izpiši »YES«, če je možno sestaviti nabor nalog, ki izpolnjujejo zgornje pogoje, in »NO«, če to ni možno.

**Razlaga vzorca podatkov**

* V prvem vzorcu, manjka lahka naloga, zato komisija ne more narediti nabor nalog z vsaj eno lahko nalogo.
* V drugem vzorcu, lahko komisija uporabi 3 lahke naloge, 7 srednje težkih nalog in 3 težke naloge, da naredi nabor nalog z natančno 13. nalogami.

Primer vhodnih podatkov 1: Primer izhodnih podatkov 1:

Text

Description automatically generated Graphical user interface, text, application

Description automatically generated

Primer vhodnih podatkov 2: Primer izhodnih podatkov 2:

A picture containing graphical user interface

Description automatically generated Text

Description automatically generated

**OPIS PROBLEMA IN IDEJA REŠEVANJA**

Vhodne podatke sestavlja vrstica štirih celih števil, ki so ločena s presledki. Prva tri števila predstavljajo število nalog pri vsaki stopnji težavnosti: lahki, srednje težki in težki. Zadnje število pa predstavlja koliko nalog mora biti skupaj. Ugotoviti moramo ali se da narediti nabor nalog, tako da je vsaj ena naloga v vsaki stopnji težavnosti.

Naloge z enako težavnostjo postavimo v ločen posamezen skupek. Preverimo, če je sploh dovolj nalog, da sestavimo nabor s takšnim številom kot je zahtevano in če je število zahtevanih nalog vsaj tri. Potem pa še preverimo, če je kakšna stopnja težavnosti manjka in če ne, so vsi zahtevani pogoji izpolnjeni ter lahko sestavimo nabor nalog za tekmovanje. Če pa niso, pa ne moremo.

**DOKAZ, DA JE NALOGA REŠENA**

AIR CONDITIONED MINIONS

Naloga je na spletni strani <https://open.kattis.com/problems/airconditioned> in je vredna 3.3 točk.



**BESEDILO NALOGE**

Si šef ACM, poštenega podjetja z enotnim ciljem popolne prevlade sveta.

Podjetje ima ***N*** minionov. Vsak minion zlobno dela od jutra do večera v super skritem bunkerju v Helsinkih. Po temeljitem premisleku, si se odločil premestiti svoj sedež v Singapur. Ampak za razliko od Helsinkov je Singapur precej toplejši, tako da mora biti celoten kompleks klimatiziran. Z napornim delovnim časom (pod minimalno plačo), je nujno da vsak minion dela v optimalnih delovnih pogojih. Še posebej pa mora vsakemu minionu odgovarjati temeperatura sobe v kateri dela.

Planiraš konstrukcijo sob v novem skrivališču, kjer bodo nameščeni tvoji minioni. Temperaturo v vsaki sobi namestiš na katero koli vrednostjo želiš (različne sobe imajo lahko različne temperature). Ko si namestil temperaturo v sobah, namestiš vsakega miniona v sobe (soba lahko vsebuje poljubno število minionov). Želiš, da bi vsem minionom ustrezala temperatura dodeljene sobe. Vsakemu minionu ustreza nek temperaturni interval in te preference ti bodo podane.

Klimatske naprave je zelo drago vzdrževati, zato želiš zgraditi karseda malo sob. Katero je minimalno število sob, ki jih moraš postaviti, da bo možno namestiti minione v sobe, na zgoraj predstavljen način?

**Vhodni podatki**

Prva vrstica vsebuje nenegativno celo število 2 ≤ N ≤ 100, ki poda število minionov v podjetju. V najslednjih ***N*** vrsticah vsaka vrstica opisuje temperaturne preference vseh minionov. Vsrstica ***i*** vsebuje dve s presledkom ločeni celi števili *L* in *U* (1 ≤ l ≤ U ≤ *2N*), kateri označujeta, da ***i***-temu minionu ustreza temperatura med *L* in *U*, vključno z *L* in *U*.

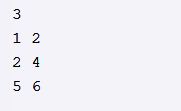
**Izhodni podatki**

Izpiši število, ki pove minimalno število sob, ki jih želiš zgraditi.

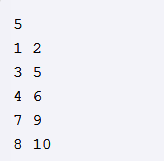
**Razlaga vzorca podatkov**

V prvem primeru je ena od možnih rešitev, da se zgradi dve sobi – prva s temperaturo 2 in druga s temperaturo 5. Prva dva miniona sta lahko nameščena v prvo sobo, tretji minion pa v drugo sobo.

Primer vhodnih podatkov 1: Primer vhodnih podatkov 1:

Primer vhodnih podatkov 2: Primer vhodnih podatkov 2:

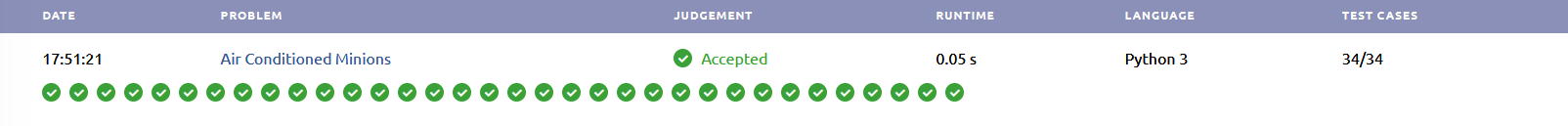
 

**OPIS PROBLEMA IN IDEJA REŠEVANJA**

Vhodne podatke sestavlja vrstica s celim številom minionov, naslednje vrstice pa vsebujejo dve celi števili ločeni s presledkom, ki predstavljajta interval temperaturne preference individualnih minionov. Ugotoviti moramo koliko sob se naredi, da bodo upoštevane vse temperaturne preference delavcev.

Vsak interval preferenc minionov shranimo v tabelo. V tabeli razvrstimo vse individualne intervale, ki so v parih po velikosti zgornjih omejitev intervalov. Potem pa po vrsti preverimo ali se intervali preferenc prekrivajo in tem minionom, katerim se interval prekriva naredimo skupno sobo. Za preostale pa individualne sobe.

**DOKAZ, DA JE NALOGA REŠENA**



BABY BITES

Naloga je na spletni strani <https://open.kattis.com/problems/babybites> in je vredna 1.9 točk.



**BESEDILO NALOGE**

Arild je ravno dopolnil 1 leto in se uči kako šteti. Najljubše mu je šteti koliko grižljajev je v obroku. Za vsak grižljaj bo naglas povedal število.

Na žalost, Arild s polnimi usti težko govori in vsake toliko časa nerazločno zamomlja. Zaradi tega je težko dognati do koliko je preštel. Včasih tudi sumiš, da izpusti kakšno število! Odločiš se, da boš napisal program, da določiš ali je Arildovo štetje smiselno ali ni.

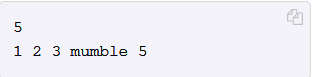
**Vhodni podatki**

Prva vrstica vsebuje celo število *n* (1 <= *n* <= 1000), ki pove koliko grižljajev je Arild prejel. Druga vrstica vsebuje *n* s presledkom ločenih besed, ki jih izgovori Arild, pri čemer je vsaka  *i*-ta ali nenegativno število *ai* (0 <= *ai* <= 10 000) ali niz »mumble«.

**Izhodni podatki**

Če je Arildovo štetje morda smiselno, izpiši niz »makes sense«. Sicer izpiši niz »something is fishy«.

Primer vhodnih podatkov 1: Primer vhodnih podatkov 1:

Primer vhodnih podatkov 2: Primer vhodnih podatkov 2:

A picture containing text

Description automatically generated A picture containing diagram

Description automatically generated

Primer vhodnih podatkov 3: Primer vhodnih podatkov 3:

A picture containing logo

Description automatically generated Text

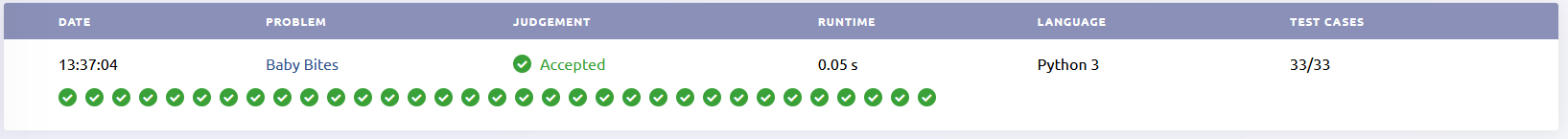
Description automatically generated with medium confidence

**OPIS PROBLEMA IN IDEJA REŠEVANJA**

Vhodne podatke sestavlja prva vrstica z enim celim številom, ki predstavlja število grižljajev, ki jih bo dojenček opravil ter druga vrstica z nizom besed, ki predstavljajo dojenčkovo štetje. Dojenček poskuša šteti vsak grižljaj. Vsake toliko časa pa nemesto štetja zamomlja. Ugotoviti se mora ali je štetje smiselno, kjer vsak momljaj obravnavamo, kot »pravilno« štetje.

Skupaj z dojenčkom štejemo vsak grižljaj. Ko zamomlja, predvidevamo, da je še vedno pravilno štel. Če se naše štetje ujema z dojenčkovim, pomeni, da je njegovo štetje smiselno. Če se ne ujema, pa ni.

**DOKAZ, DA JE NALOGA REŠENA**



BOSS BATTLE

Naloga je na spletni strani <https://open.kattis.com/problems/bossbattle> in je vredna 1.8 točk.



**BESEDILO NALOGE**

Obtičal si pri 'boss' stopnji svoje najljubše video igre. 'Boss' bitka poteka v okrogli sobi z *n* neuničljivimi stebri, ki so enakomerno razporejeni. 'Boss' se skriva za neznanim stebrom. Z 'Bossom' izmenjaje nadaljujeta.

* Prvič, ko si ti na vrsti lahko vržeš bombo za enega od stebrov. Bomba bo premagala 'bossa', če je za tem ali za sosednjima stebroma.
* Nato, če 'boss' ni bil premagan, lahko, ko je na vsrti, ostane kjer je ali pa se premakne za enega od njemu sosednjih stebrov. Dim, ki ga povzroči bomba zakriva premike 'bossa'.

Ko si nazadnje poiskusil premagati 'bossa', ti je spodletelo, ker ti je zmanjkalo bomb. Tokrat želiš zbrati dovoj bomb, da karkoli 'boss' naredi, ga boš premagal. Kolikšno je minimalno število bomb, ki jih potrebuješ, da v najslabšem primeru premagaš 'bossa'? Poglej sliko 1 za primer.

A picture containing clock

Description automatically generated

prva bomba po eksploziji po premiku 'bossa' druga bomba

**Slika 1**: Primer za *n* = 4. V tem primeru sta 2 bombi dovolj. Sivi stebri predstavljajo stebre,

za katere se 'boss' ne more skriti. Bomba je predstavljena s črno.

**Vhodni podatki**

Vhodni podatki vsebujejo:

* Ena vrstica z enojnim celim številom n (1 <= n <= 100), ki predstavlja število stebrov v sobi.

**Izhodni podatki**

Izpiše naj minimalno število potrebnih bomb, da premaga 'bossa' v najslabšem primeru.

Primer vhodnih podatkov 1: Primer izhodnih podatkov 1:

A picture containing rectangle

Description automatically generated Rectangle

Description automatically generated with medium confidence

Primer vhodnih podatkov 2: Primer izhodnih podatkov 2:

Shape, rectangle

Description automatically generated Rectangle

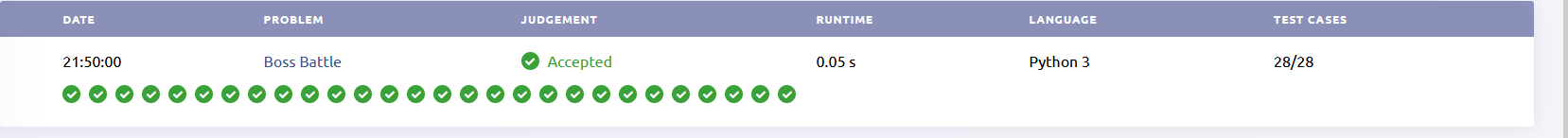
Description automatically generated with low confidence

**OPIS PROBLEMA IN IDEJA REŠEVANJA**

Edini vhodni podatek je celo število, ki predstavlja število stebrov v krožni sobi. Da premagaš bossa moraš vreči bombo za steber in če stoji za tem ali sosednjima, je premagan. Ugotoviti moramo način kako neizogibno premagati bossa, potem pa kaj je najslabši primer tega načina ter točno koliko bomb je potrebnih za ta primer.

Edini način, da vedno premagaš bossa, je da vržeš bombo za naključni steber, potem izbereš smer in mečeš bobme po vrsti za vsaki drugi steber. Če bi vrgli za vsak ali vsak tretji steber, bi se boss vedno lahko izmuznil. Najslabši primer tega pa je, da smo po prvem metu za en steber zgrešili bossa, zatem pa nam je boss sledil. Ker smo za en korak hitrejši, ga vedno ujamemo. Ugotovimo, da je število minimalno potrebnih bomb vedno dva manj od števila stebrov. Edina izjema je takrat, ko je samo en steber ali dva stebra, kjer je minimalno število potrebnih bomb ena.

**DOKAZ, DA JE NALOGA REŠENA**



SIMON SAYS

Naloga je na spletni strani <https://open.kattis.com/problems/simon> in je vredna 3.0 točk.



**BESEDILO NALOGE**

'Simon reče' je igra kjer je eden igralec Simon, drugi pa morajo narediti to kar reče, ampak samo če začne stavek z »Simon says«. Če ti ne uspe izvesti ukaza, izgubiš. Če izvedeš ukaz, ki se ni začel s čarobnimi besedami, tudi izgubiš. Igraš že nekaj časa in postaja malo duhamorno, ker je prelahko. Napiši program, ki ti pomaga odločiti se, kaj narediti glede na Simonov ukaz.

**Vhodni podatki**

Prva vrstica naj vsebuje celo število T, ki predstavlja število testnih primerov. Vsak testni primer je sestavljen iz ene vrstice besedila – Simonov ukaz.

* 1 <= T <= 20
* Vsaka vrstica vsebuje samo črke a-z in presledke
* Vsaka beseda bo ločena z natančno enim presledkom
* Vrstice ne bodo imele presledke pred stavkom in po stavku
* Vsaka vrstica bo imela med 1 in 1000 znakov

**Izhodni podatki**

Za vsak testni primer ponovi ukaz, če sta bile prve dve besedi »simon says«. Ne ponovi začetni besedi »simon says«. Drugače izpiši prazno vrstico.

Primer vhodnih podatkov 1: Primer izhodnih podatkov 1:Text, letter

Description automatically generated Graphical user interface, text

Description automatically generated

**OPIS PROBLEMA IN IDEJA REŠEVANJA**

Za vhodni podatek dobiš število testnih primerov, ki jih moraš izvesti. Testni primer je sestavljen iz besed ločenih s presledkom. Vsak testni primer se lahko začne s »simon says« lahko pa ne. Napisati moramo program, kjer z danimi ukazi, izpiše ukaz nazaj, če je pogoj, da se začne z besedama »simon says« izpolnjen. Če ni, vrne prazno vrstico.

Preberemo število potrebnih testnih primerov in vsak testni primer posebej preverimo, če se začne s »simon says«, če se, izpišemo ukaz nazaj brez prvih 11 znakov, kar točno pride ukaz brez prvih dveh besed »simon says«. To ponovimo tolikokrat, kot je bilo število testnih primerov dano.

**DOKAZ, DA JE NALOGA REŠENA**



MILLIONAIRE MADNESS

Naloga je na spletni strani <https://open.kattis.com/problems/millionairemadness> in je vredna 2.6 točk.



**BESEDILO NALOGE**

Tvoj bližnji prijatelj, racak s finančnimi problemi, vas je prosil za pomoč pri zadevi, ki mu bo pomagala odplačati dolgove. Je nečak zelo bogatega racaka, ki ima velik trezor napolnjen z gorami kovancev. Bogati racak ima kovanec, ki ima zanja veliko sentimentalno vrednost. Ponovadi je shranjen pod zaščitno stekleno kupolo na žametni blazini.

Vendar je bil poseben kovanec med nedavno premestitvijo kovancev v trezorju pomotoma premaknjen v trezor, kar je pripeljalo strica tvojega prijatelja v izjemno stresno situacijo. Na srečo je bil kovanec nedavno lociran. Na žalost pa je kovanec popolnoma nasproti vhoda trezorja in zaradi gore kovancev v trezorju ni lahka naloga priti do kovanca.

Tvojemu prijatelju je pripravljen plačati, da prinese kovanec, pod pogojem, da pripelje svojo opremo s katero bo preplezal gore kovancev. Tvoj prijatelj se je odločil, da bo prinesel lestev, ampak ni gotov glede njene višine. Daljša lestev namreč pomeni, da lahko prepleza večje kupe kovancev, a ga tudi več stane. Zato hoče kupiti nakrajšo lestev s katero bo dosegel posebni kovanec ter da mu pri tem ostane kar največ denarja za povračilo dolgov.

Trezor je lahko predstavljen kot kvadrat koordinatne mreže s kupi kovancev različnih višin (v metrih), z vhodom na severno zahodnem kotu (prva višina v vhodnih podatkih, vhod v trezor je na isti višini), poseben kovanec pa je pri južno vzodnem kotu (zadnja višina v vhodnih podatkih). Tvoj ptičji prijatelj je pogruntal višine kovancev v vsakem od kvadratov. S kupa kovancev lahko poskusi plezati navzgor ali skočiti s kupa na kup, ki je neposredno severno, južno ali vzhodno od njega. Ker tvoj prijatelj ne more ne skočiti ne leteti (je zelo posebne vrste racak, ki tudi nosi obleke), bi moral prinesti lestev dolžine vsaj n metrob, da bi lahko uspešno preplezal višino n metrov. Ne moti ga skok navzdol ne glede s katere višine, saj le pusti gravitaciji, da opravi svoje.

**Vhodni podatki**

Prva vrsta vsebuje dve celi števili: dolžino M, in širino N trezorja, ki zadovoljita 1 ≤ M, N ≤ 1000.

Vse naslednj M vrstice vsebujejo N cela števila. Vsako celo število pove višino kupa kovancev na določeni poziciji. (Prva vrstica opisuje najbolj severne kupe od zahoda proti vzhodu; zadnja vrstica pa opisuje najbolj južne kupe od zahoda proti vzhodu). Višine so dane v metrih in vse višine so vsaj 0 do njaveč (da, tvoj prijatelj ima zelo bogatega strica).

**Izhodni podatki**

Izpiši eno vrstico, ki vsebuje eno samo celo število: dolžina v metrih najkrajše lestve, ki ti omogoči iti iz severno zahodnega kota do južno vzhodnega kota.

Primer vhodnih podatkov 1: Primer izhodnih podatkov 1:

Text

Description automatically generated with low confidence A picture containing icon

Description automatically generated

Primer vhodnih podatkov 2: Primer izhodnih podatkov 2:

Graphical user interface

Description automatically generated with medium confidence Graphical user interface, text, application, chat or text message

Description automatically generated

Primer vhodnih podatkov 3: Primer izhodnih podatkov 3:

Text

Description automatically generated Graphical user interface, text, application, chat or text message

Description automatically generated

**OPIS PROBLEMA IN IDEJA REŠEVANJA**

Vhodne podatke sestavljajo vrstica z dvema številoma, ki predstavljata dolžino in širino trezorja, naslednje vrstice pa vsebuje števila z višinami vseh kupov kovancev v trezorju. Najti moramo pot skozi trezor, ki ima najmanjši maksimum razlik med višinami kupov.

Preberemo informacije o dimenziji sobe, ter višine kupov kovancev. Začnemo zgoraj levo in pogledamo višine možnih sosednjih kupčkov ter si zapišemo razlike višin ter njihove koordinate v prioritetnem vrstnem redu, kjer je prvi na seznamu vedno tisti kup z najmanjšo razliko. Prečrtamo trenutne koordinate, da se ne vračamo na isto mesto ter shranimo največjo razliko višin na trenutni poti. Potem pa izberemo prvi kup na seznamu in se premaknemo tja. Uporabljamo prioritetni seznam, kar pomeni, da če imajo na trenutni poti vsi sosednji kupi večjo razliko višin kot sosednji kupi druge poti, izberemo vedno pot z manjšo razliko v tem primeru drugo pot. Te korake ponavljamo, dokler ne pridemo do konca, ki je spodaj desno. Ko pridemo do konca, pogledamo najvišjo razliko višin na tej poti ter jo izpišemo. Ta višina je dolžina želene lestve.

**DOKAZ, DA JE NALOGA REŠENA**

Poročilo 2.del

Guessing game

Naloga je na spletni strani <https://open.kattis.com/problems/guessinggame> in je vredna 3.1 točk.



**BESEDILO NALOGE**

Stan in Ollie igrata igro ugibanja. Stan si zmisli število med 1 in 10 (vključeno) in Ollie ugiba katero število bi lahko bilo. Po vsakem ugibanju, Stan namiguje ali je Ollijevo ugibanje previsoko, prenizko ali točno.

Po nekaj rundah, Ollie začne sumiti, da Stan goljufa, to je, da spreminja svoje število med Ollijevimi ugibanji. Za pripravo postopka zoper Stana, je Ollie beležil zapise nekaj iger. Presoditi moraš ali posamezen zapis dokazuje Stanovo goljufanje.

**Vhodni podatki:**

Vhodni podatki vsebujejo nekaj zapisov iger. Vsak zapis je sestavljen iz števila parov ugibanj in odgovorov. Ugibanje je vrstica, ki vključuje samo celo število med 1 in 10 (vključeno) in odgovor z vrstico, ki vsebuje niz »too high«, »too low« ali »right on«. Vsaka igra se zaključi z nizom »right on«. Vrstica z 0 sledi zadnjemu zapisu. Skupno število parov ugibanje-odgovor je največ 2500.

**Izhodni podatki:**

Za vsako igro, izpiši vrstico z nizom »Stan is dishonest«, če so Stanovi odgovori nekonsistentni z zadnjim ugibanjem in odgovorom. Sicer izpiši niz »Stan may be honest«.

Primer vhodnih podatkov 1: Primer izhodnih podatkov 1:

Graphical user interface

Description automatically generated with medium confidence Graphical user interface, text

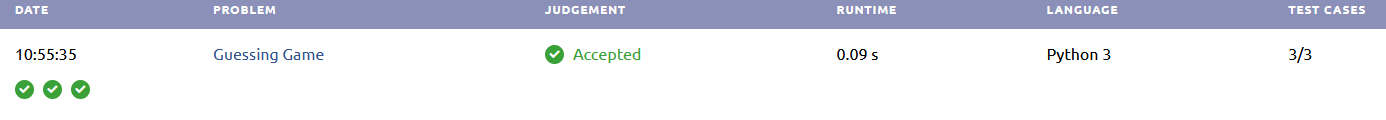
Description automatically generated

**OPIS PROBLEMA IN IDEJA REŠEVANJA**

Vhodne podatke sestavljajo pari vrstic s celim številom, ki predstavlja trenutno ugibanje Ollija in z nizom, ki predstavlja odgovor Stana na ugibanje. Zadnja vrstica vsebuje število 0, ki označuje konec igre. Ugotoviti moramo ali Stan goljufa tako da spreminja število med ugibanji.

Preberemo zapis in shranimo v pare število trenutnega ugibanja in odgovor na to ugibanje. Zapišemo spodnjo in zgornjo mejo intervala, možnih rešitev igre. Na začetku runde je interval med 1 in 10. Gremo skozi zapis in ko je Stan odgovoril »too low« na Ollijevo ugibanje, posodobimo spodnjo mejo s številom, ki je večje za eno od Ollijevega. Pred tem pa še primerjamo trenutno vrednost z novo in posodobimo, če je nova večja od trenutne. To naredimo zato, da interval ostane nespremenjen v primeru, da Stan goljufa in spremeni število med rundo igre. Če je Stan odgovoril »Too high«, pa obratno; zgornjo mejo posodobimo s številom, ki je za eno manjše od trenutnega ugibanja, v primeru, da je to število večje od trenutne vrednosti, ostane zgornja meja nespremenjena. To ponavljamo, dokler ne pridemo do »pravilnega odgovora«. Če je to število v našem intervalu, ki smo ga posodabljali, Stan mogoče ni goljufal, če ni v intervalu, pa je defenitivno goljufal. Preverimo še preostale runde do konca igre.

**DOKAZ, DA JE NALOGA REŠENA**



Knight Jump

Naloga je na spletni strani <https://open.kattis.com/problems/knightjump> in je vredna 2.4 točk

**BESEDILO NALOGE**

Dobil si dvodimenzionalno šahovnico velikosti N x N (1-osnovno indeksiranje). Nekatere celice na šahovnici so  **̒ . ̓**, ki predstavljajo prazno celico. Nekatere celice pa so  **̒ # ̓** , ki predstavljajo blokirano celico, ki je ne smeš obiskati. Natančno ena izmed celic na šahovnici je **̒ K ̓** , ki predstavlja začetni položaj viteza.

Vitez na mestu (r, c), se lahko premakne na katerokoli veljavno mesto v množici S = {(r + 2, c + 1), (r + 2, c - 1), (r - 2, c + 1), (r - 2, c - 1), (r + 1, c + 2), (r + 1, c - 2), (r - 1, c + 2), (r - 1, c - 2)}. Veljavno mesto tu pomeni, da mora (r', c') biti znotraj meje šahovnice, t, j. 1 ≤ r '≤ N in 1≤ c '≤ N.

Vprašanje je, kolikšno je minimalno število potrebnih premikov, da vitez pride do celice (1, 1), s tem, da se izogiba celicam z ̒ # ̓ na poti.

Opomba – na mreži bo natanko en  **̒ K ̓**  in celica (1,1) ne bo  **̒ # ̓** .

**Vhodni Podatki**

Prva vrstica vsebuje celo število N (1 ≤ N ≤ 102), ki predstavlja dimenzijo šahovnice. Vsaka naslednja N vrstica vsebuje niz, ki predstavlja i-to vrstico. Dolžina tega niza bo N.

**Izhodni Podatki**

Izpiši vrednost minimalno potrebnih premikov. Vendar, če (1,1) ni možno doseči, izpiši  **̒ -1 ̓** (brez narekovajev).

Primer vhodnih podatkov 1: Primer izhodnih podatkov 1:

A picture containing table

Description automatically generated Graphical user interface, text

Description automatically generated

Primer vhodnih podatkov 2: Primer izhodnih podatkov 2:

A picture containing application

Description automatically generated Chart, box and whisker chart

Description automatically generated

**OPIS PROBLEMA IN IDEJA REŠEVANJA**

Vhodne podatke sestavlja vrstica s celim številom N, ki predstavlja dimenzijo šahovnice n\*n, sledijo vrstice z nizi, ki vsebujejo znake:  **̒ . ̓ , ̒ # ̓** in **̒ K ̓**, kjer  **̒ . ̓** predstavlja mesto na šahovnici, ki ga lahko obiščemo,  **̒ # ̓** predstavlja mesto na šahovnici, ki ga ne smemo obiskati ter  **̒ K ̓**, ki predstavlja naš začetni položaj. Premikamo se lahko samo tri mesta v eno smer in na koncu tretjega premika lahko zavijemo levo ali desno za en položaj. Se pravi, lahko se premikamo v L-jih kot figura konja v igri šaha. Ne smemo se premakniti na mesto, ki vsebuje **̒ # ̓**. Naš cilj je, da z najmanjšim številom potez pridemo do cilja, ki je vedno na mestu (0,0). Če ne moremo priti do cilja, izpišemo -1.

Vrstice šahovnice shranimo posebej v tabelo. Poiščemo in izberemo začetni položaj. Preverimo, če smo že na cilju. Če nismo, poskusimo vsako možno potezo tega položaja. Vsako potezo posebej preverimo, če nas pripelje izven meje šahovnice, če je bil položaj že obiskan ali pa če položaj vsebuje **̒ # ̓**. Če je poteza pravilna, položaj skupaj s posodobljenim številom opravljenih potez shranimo v naš seznam. Kjer je prvi dodan seznamu, je tudi prvi, ki bo izbran iz seznama. Položaj dodamo k že obiskanim. Ponovimo za vsako pravilno potezo. Po tem pogledamo naš seznam in izberemo nov položaj in število potez, ki sta bila prva dodana. Ponovimo vse korake dokler ne pridemo do cilja ali pa če cilj ni dosegljiv.

**DOKAZ, DA JE NALOGA REŠENA**



EGYPT

Naloga je na spletni strani <https://open.kattis.com/problems/egypt> in je vredna 2.1 točk



**BESEDILO NALOGE**

Dolgo časa nazaj so Egipčani ugotovili, da je največji kot pri trikotniku s stranicami dolžine 3, 4 in 5 vedno pravi kot. Ugotoviti moraš ali imajo drugi trikotniki podobno lastnost.

**Vhodni Podatki**

Vhodni podatki predstavljajo nekaj testnih primerov (največ 1000), sledi vrstica s tremi ničlami 0 0 0. Vsak testni primer ima tri pozitivna cela števila, največ 30 000, ki predstavljajo dolžine stranic trikotnika.

**Izhodni Podatki**

Za vsak testni primer, izpiši vrstico, ki vsebuje »right«, če je trikotnik pravokoten in »wrong«, če trikotnik ni pravokoten.

Primer vhodnih podatkov 1: Primer izhodnih podatkov 1:

A picture containing text

Description automatically generated Text

Description automatically generated with medium confidence

**OPIS PROBLEMA IN IDEJA REŠEVANJA**

Vhodne podatke sestavljajo vrstice, ki vsebujejo dolžine stranic trikotnikov, zadnja vrstica pa vsebuje tri ničle. Ugotoviti moramo ali so trikotniki pravokotni in če so, vrnemo niz »right«, če pa niso, vrnemo niz »wrong«.

Shranimo dolžine stranic v pare treh. Vsak par uredimo po velikosti, da je najdaljša stranica vedno zadnja. Potem za vsak trikotnik izračunamo hipotenuzo s pomočjo Pitagorovega izreka »c² = a² \* b²« tako, da manjši stranici kvadriramo, seštejemo ter korenimo. Če je izračun enak naši najdaljši stranici vemo, da je trikotnik pravokoten.

**DOKAZ, DA JE NALOGA REŠENA**



COLLATZ CONJECTURE

Naloga je na spletni strani <https://open.kattis.com/problems/collatz> in je vredna 4.2 točk



**BESEDILO NALOGE**

Collatzova domneva je zanimiv fenomen. Čeprav je princip zelo preprost, še vedno ostaja eden izmed nerešenih problemov v matematiki, tudi po mnogih letih reševanja. Ampak leta intenzivnega raziskovanja so prinesla kar nekaj razultatov, kar je velik napredek človeške rase proti vesoljcem, zato ker niso raziskovali domneve toliko let. To prednost hočemo obdržati.

Zamisli si zaporedje definirano rekurzivno kot sledi: začneš z katerim koli celim številom x0 (imenovan »začetna vrednost«). Potem ponavljaj sledeče:

* Čeje xi sodo število, potem xi +1 = xi/2 (»polovica…«)
* Če je xi liho število, potem x+1 = 3xi + 1 (»…ali trikratnik plus ena«)

Collatzova domneva pravi, da bo vsako tako zaporedje naposled doseglo število 1. Še vedno ni dokazano, ampak mi že zagotovo vemo, da to velja za vse x0 < 258. (Tega nikoli ne povej vesoljcem!)

V tem problemu sta dani dve začetni vrednosti in tvoja naloga je povedati v kolikih korakih se njuno zaporedji »srečaza« prvič (kar pomeni *prvo* število, ki se pokaže pri obeh zaporedjih) in pri katerem številu se bosta srečala. Zaradi poenostavitve, bomo predpostavili, da se zaporedje ne nadeljuje, ko doseže število ena. V realnosti bi se to spremenilo v 1, 4, 2, 1, 4, 2, 1,…, kar hitro postane dolgočasno.

**VHODNI PODATKI**

Vhodni podatki vsebujejo največ 1 500 testov. Vsak test je opisan z eno samo vrstico, ki vsebuje dve celi števili A in B, 1 ≤ A, B ≤ 1 000 000.

Zadnjemu testu sledi vrstica, ki vsebuje dve ničli.

**IZHODNI PODATKI**

Za vsak test, izpiši stavek »A needs Sa steps, B needs Sb steps, they meed at C«, kjer sta Sa in Sb število potrebnih korakov, da obe zaporedji dosežeta isto število C. Natančno se drži formata izpisa.

Primer vhodnih podatkov 1: Primer izhodnih podatkov 1:

A picture containing table

Description automatically generated Graphical user interface, text

Description automatically generated Text

Description automatically generatedText

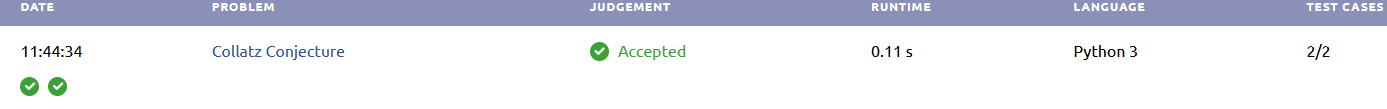
Description automatically generated

**OPIS PROBLEMA IN IDEJA REŠEVANJA**

Vhodne podatke sestavljajo vrstice, ki vsebujejo dve števili. Zadnja vrstica vsebuje števili 0 0. Najti moramo enako število v Collatzovem zaporedju dveh števil ter izpisati koliko korakov je bilo potrebnih narediti, da smo prišli do iskanega števila pri obeh zaporedjih.

Vsa števila shranimo v pare. Pri vsakem paru prvo izračunamo Collatzovo zaporedje pri prvem številu ter shranimo vsak člen zaporedja s svojim indeksom v slovar. Potem pa začnemo izračunavati Collatzovo zaporedje drugega števila in za vsak izračunan člen preverimo, če je v našem slovarju. Ko pridemo do takega števila, shranimo to število skupaj z indeksom drugega zaporedja. In izpišemo koliko korakov je bilo potrebnih, da smo prišli do iskanega števila pri obeh zaporedjih. To ponovimo za vsak par.

**DOKAZ, DA JE NALOGA REŠENA**



RATIONAL ARITHMETIC

Naloga je na spletni strani <https://open.kattis.com/problems/rationalarithmetic> in je vredna 3.5 točk

**VHODNI PODATKI**

Prva vrstica vhodnih podatkov vsebuje eno celo število z navedbo števila potrebnih matematičnih operacij.

Zatem sledijo matematične operacije, ena na vrstico, vsaka z obliko x1 y1 op x2 y2. Pri tem so -109 <= x1,y1,x2,y2 < 109 cela števila, ki kažejo da so operandi x1/y1 in x2/y2. Operator op je eden izmed »**+**«, »**-**», »**\***«, »**/**«, ki kažejo na matematično operacijo, ki jo moramo izvesti.

Predpostaviš lahko, da y1 ≠ 0, y2 ≠ 0 in da je x2 ≠ za operacijo deljenja.

**IZHODNI PODATKI**

Za vsako operacijo v vsakem testnem primeru izpiši rezultat nakazane operacije v najkrajši prikazani obliki.

Primer vhodnih podatkov 1: Primer izhodnih podatkov 1:

Text

Description automatically generated Text

Description automatically generated

**OPIS PROBLEMA IN IDEJA REŠEVANJA**

Za vhodni podatek dobimo število primerov, ki jih bomo morali rešiti. Vsak primer v svoji vrstici vsebuje dva para celih števil, ki so ločena s presledki, kjer presledek predstavlja ulomek, pari so pa ločeni z računsko operacijo. Moramo izvesti računsko operacijo z danimi ulomki.

Individualne primere shranimo v tabelo ter postavimo računsko operacijo na konec te tabele. Pri vsakem primeru pogledamo katero računsko operacijo vsebuje, v primeru deljenja obrnemo števec in imenovalec drugega ulomka ter zmnožimo oba števca in imenovalca med seboj, pri množenju naredimo enako, samo brez prvega koraka, pri seštevanju in odštevanju pa najprej pogledamo, če imata oba ulomka enak imenovalec, če nimata, zmnožimo imenovalca skupaj ter pomnožimo vsak števec z nasprotnima imenovalcema in odštejemo ali seštejemo dobljena števca skupaj. Pri vsakem primeru pokrajšamo ulomek tako, da preverimo, če imata imenovalca skupni delitelj, ki je večji od 1. Na koncu preverimo še, če je imenovalec negativen. Če je, zamenjamo predznak s števcem, v primeru ko pa sta oba negativna se znebimo obeh predznakov.

**DOKAZ, DA JE NALOGA REŠENA**

**POROČILO 3. DEL**

INVERSE FACTORIAL

Naloga je na spletni strani <https://open.kattis.com/problems/inversefactorial> in je vredna 5.8 točk



**BESEDILO NALOGE**

Fakulteta n! pozitivnega celega števila n je definiran, kot produkt vseh pozitivnih celih števil, ki so manjše ali enaki n-ju. Primer,

21! = 1 \* 2 \* 3 \* … \* 21 = 51 090 942 171 709 440 000.

Preprosto je izračunati fakulteto majhnega celega števila in verjetno so to storil velikokrat prej. V tem problemu je tvoja naloga obrnjena. Dobiš razultat n! in moraš ugotoviti vrednost n-ja.

**VHODNI PODATKI**

Vhodni podatki vsebujejo fakulteto n! pozitivnega celega števila n. Število števk n! je največ 10^6.

**IZHODNI PODATKI**

Vrni vrednost n-ja.

Primer vhodnih podatkov 1: Primer izhodnih podatkov 1:

Graphical user interface

Description automatically generated with low confidence Graphical user interface

Description automatically generated with low confidence

Primer vhodnih podatkov 2: Primer izhodnih podatkov 2:

Text

Description automatically generated with low confidence Graphical user interface, text

Description automatically generated with medium confidence

Primer vhodnih podatkov 3: Primer izhodnih podatkov 3:

Text

Description automatically generated with medium confidence Graphical user interface, text, application

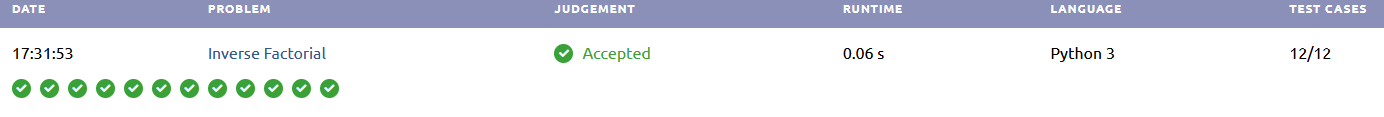
Description automatically generated

**OPIS PROBLEMA IN IDEJA REŠEVANJA**

Za vhodni podatek dobimo eno celo število, ki predstavlja vrednost n fakultete neznanega števila. Ugotoviti moramo katero je to število. Paziti moramo tudi, da je časovna omejitev CPU-ja 1 sekunda.

Ker računanje n fakultete porabi veliko časa, tudi s pomočjo rekurzije, uporabimo lastnost logaritmov in sicer : log(a\*b) = log(a) + log(b). Izberemo logaritem z osnovo 10, ker tako lahko izvemo koliko števk je pri vsaki n fakulteti (če zaokrožimo navzgor). Ta metoda deluje samo s števili, ki imajo več kot eno števko, zato najprej preverimo, če je število eden izmed 1!, 2! ali pa 3!. Če ni začnemo seštevati logaritme od 1 naprej, skupaj pa štejemo kolikokrat smo sešteli. Ko je vsota logaritmov večja od števila števk našega števila, vrnemo število seštevkov, ki pa je naše iskano število.

**DOKAZ, DA JE NALOGA REŠENA**



FALLING MUGS

Naloga je na spletni strani <https://open.kattis.com/problems/falling> in je vredna 4.0 točk

**BESEDILO NALOGE**

Susan ustvarja visoko-hitrostne posnetke padajočih kozarcov. Med analiziranjem posnetkov je hotela vedeti kako velki so kozarci ampak na žal so bili vsi uničeni med snemanjem. A Susan pozna malo fizike, ugotovila je kako dolgo so padali med različnimi video sličic. Kamera je snemala 70 sličic na sekundo, kar pomeni da pri sličici n s štetjem ko je bil kozarec spuščen se je premaknil za d = n\*\*2 milimetrov. Štetje sličic se začne pri 0.

Susan misli, da je določen kozarec D milimetrov visok, ampak, da to dokaže mora najti dve sličici med katerimi se je premikal kozarec točno te razdalje. Ji lahko pomagaš to narediti?

VHODNI PODATKI

Vhodni podatki vsebujejo samo pozitivno celo število D <= 200 000, ki predstavlja razdaljo ki jo je treba izmeriti.

IZHODNI PODATKI

Izpiši dve ne negativni celi števeli n1 in n2, število sličic ki jih mora susan primerjati. Izpolniti morajo n2\*\*2 – n1\*\*2 = D. Če te dve celi števili ne obstajata izpiši »impossible«.

Primer vhodnih podatkov 1: Primer izhodnih podatkov 1:

Graphical user interface, text, application

Description automatically generated Graphical user interface, text, application

Description automatically generated

Primer vhodnih podatkov 2: Primer izhodnih podatkov 2:

Graphical user interface, application

Description automatically generated with medium confidence Text

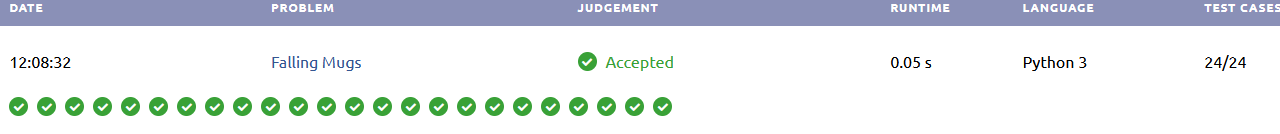
Description automatically generated with medium confidence

**OPIS PROBLEMA IN IDEJA REŠEVANJA**

Za vhodni podatek dobimo eno celo število, ki predstavlja prepotovano razdaljo skodelice med dvema sličicema. Z drugimi besedami visino skodelice. Ugotoviti moramo ali je podana razdalja med dvema sličicama možna, če upoštevamo dani pogoj n2\*\*2 – n1\*\*2 = D, kjer je n1 prva sličica, n2 druga sličica in D razdalja. Če ni možna izpišemo »impossible«.

Razstavimo razliko kvadratov a\*\*2 – b\*\*2 = D in dobimo (a+b)(a-b) = D, kjer je (a-b) <= (a+b). Gremo skozi vse možne faktorje od 1 do √D (vključno √D), ki bi nadomestili (a-b). Za vsakega preverimo če je deljenec D deljiv s tem faktorjem, to je prvi pogoj, ki mora biti izpolnjen. Da najdemo drugi možni faktor (a+b) delimo D s prvim možnim faktorjem (a-b). Potem preverimo če sta oba možna faktorja liha ali soda, to je drugi pogoj, ki mora biti izpoljen. Ker sta sestavljena z istimi spremenljivkami morata oba biti soda ali liha. (če je a-b sodo je tudi a+b sodo). Če sta oba pogoja izpolnjena smo našli rešitev. Izračunamo ven spremenljivki a in b. Če pa pogoja nista bila izpolnjena vemo, da sistem nima rešitev. (v množici ℤ)

**DOKAZ, DA JE NALOGA REŠENA**



VIRI

Air Conditioned minions

V petnajsti vrstici programa airconditionedminions.py vir kode: »preference.sort(key = lambda druga\_pref: druga\_pref[1])«

<https://www.geeksforgeeks.org/python-sort-list-according-second-element-sublist/>

Symon says

Vir celotne kode: <https://github.com/KentGrigo/Kattis/blob/master/python/simonsays.py>

Millionare Madness

Vir ideje kode:

Dijkstra's Shortest Path Algorithm | Graph Theory <https://www.youtube.com/watch?v=pSqmAO-m7Lk>

Knight Jump

Python Path Finding Tutorial - Breadth First Search Algorithm - <https://www.youtube.com/watch?v=hettiSrJjM4> (ideja za kodo)