Leták zimnej časti III. ročníka

Ahojte milí riešitelia.

Sme veľmi radi, že ste sa dozvedeli o PRASKu a asi by vás zaujímalo, čo to vlastne je, ako to celé funguje a prečo by ste to mali riešiť. Na všetko z toho sa vám teraz pokúsime odpovedať.

Čo to je a pre koho je to určené?

PRASK je korešpodenčný seminár určený pre všetkých základoškolákov, ktorých zaujíma matematika, informatika alebo by sa chceli naučiť programovať. Je to súťaž zameraná hlavne pre siedmakov a starších, môžete ju však riešiť aj keď ste v nižšom ročníku.

Seminár je organizovaný študentami informatiky na Fakulte matematiky, fyziky a informatiky na Univerzite Komenského.

Priebeh súťaže

Počas roka prebiehajú dve nezávislé časti – letná a zimná. Priebeh častí je už potom úplne rovnaký. Každá časť pozostáva z dvoch sérií piatich príkladov – dvoch teoretických, jedného praktického a dvoch programátorských. Ak aj neviete programovať nezúfajte. Namiesto programátorských úloh si môžete prejsť programátorským tutoriálom, ktorý vás to naučí a naviac v ňom získate body, ktoré sa vám rátajú do PRASKu.

Na riešenie série je vyhradených niekoľko týždňov. Až do dňa odovzdania môžete doma riešiť zadané príklady. Môžete riešiť ľubovoľné príklady z danej série, nemusíte vyriešiť všetko, nemusíte vyriešiť ani celú úlohu[1]. Najneskôr do dňa odovzdania (ktorý je napísaný na zadaniach aktuálnej série) je potrebné poslať vaše riešenia pomocou webového rozhrania.

Po konci série si pozrieme vaše odovzdané riešenia a opravíme ich. Pre každý príklad je v zadaní napísané, koľko bodov sa zaň dá dostať. Samozrejme, je možné získať čiastkové body, aj keby ste nevyriešili celú úlohu, alebo by vaše riešenie nebolo úplne správne. Dokonca, ak nás prekvapíte originálnym riešením, môžete získať bonusové body. Opravené riešenie vám potom pošleme späť aj s poznámkami ohľadom vašeho riešenia.

Prečo to chcem riešiť?

Riešenie korešpodenčného seminára prináša mnoho výhod. Riešením úloh a čítaním našich vzorových riešení **objavíte a naučíte sa** mnoho nových vecí, ktoré by ste sa v škole skoro určite nenaučili. Napríklad sa môžete naučiť **programovať**. To vám potom vie **pomôcť pri prijímačkách**, či už na stredné alebo vysoké školy. Takisto vám to pomôže pri **riešení Olympiády z informatiky alebo Korešpodenčného Seminára z Programovania**. No a v neposlednom rade, pri **pohovoroch** do veľkých firiem ako Google, Facebook alebo Eset častokrát zaváži znalosť algoritmického programovania, ktoré si môžete pomocou nášho semináru trénovať.

Je tu však ešte jedna výhoda určená pre najlepších riešiteľov. Dvakrát ročne sa bude organizovať týždenné **sústredenie**. Naň pozývame niekoľko[2] najlepších riešiteľov. Na sústredení si užiješ kopec zábavy, športu, nových ľudí a možno sa aj niečo naučíš.

A samozrejme, víťazov čakajú pekné **vecné ceny** vo forme knižky, hry alebo menšej elektroniky.

Ako má vyzerať správne riešenie

To závisí od typu úlohy, ktorú riešite. Pri teoretických úlohách musí správne riešenie okrem výsledku obsahovať aj popis postupu, akým ste sa k danému výsledku dopracovali. Dôraz sa pri opravovaní dáva hlavne na tento slovný popis, ktorý by mal byť napísaný čo najzrozumiteľnejšie, aby sme si pri opravovaní nemuseli lámať hlavu. Mal by obsahovať všetky podstatné kroky, ktoré vás viedli k riešeniu.

[1] Aj keď budeme radi, ak sa vám to podarí.

[2] zhruba 15, ale aj nižšie umiestení riešitelia sa môžu dostať ako náhradníci

V prípade praktických úloh sa to líši. Občas od vás chceme slovný popis, občas sa stačí dostať k nejakému tajnému heslu alebo kliknúť na správnu linku. Presný spôsob nájdete v zadaní.

No a pri programátorských úlohách a programátorskej liahni odovzdávate iba váš program, ktorý sa vám okamžite automaticky otestuje a do pár sekúnd sa dozviete, či ste úlohu vyriešili správne. A ak nie, môžete skúsiť odovzdať opravený program znova.

A nebojte sa, ak ste ešte nikdy nespisovali postupy svojich riešení. Keď vám riešenia opravíme, napíšeme vám k nim aj komentáre, ktoré vám môžu pomôcť v riešení ďalšej série. To je najlepší spôsob, ako sa zlepšovať.

Spôsob odovzdávania

Ako prvú vec, ktorú musíte urobiť pred tým, ako budete môcť odovzdávať svoje riešenia, je **zaregistrovanie** sa na našej webovej stránke `prask.ksp.sk`. V časti `Zadania a vzoráky` nájdete okrem zadaní aj odkaz, na ktorom môžete odovzdať vaše riešenie.

Riešenie každej teoretickej úlohy má byť jeden súbor formátu **.pdf**. Ten nahráte na našu stránku a stlačíte zelené tlačítko `Submit`. Opravovať sa bude **posledné odovzdané** riešenie, takže si dajte pozor, aby ste si niečo neprepísali.

Myslím, že vytvoriť pdf súbor by pre vás nemal byť problém, ak by ste s tým predsa len problém mali, pokúste sa použiť nejaký online converter ako napríklad `www.freepdfconvert.com`.

V prípade programátorských úloh sa dá rovnakým spôsobom odovzdať zdrojový kód vášho programu, teda súbor s príponou **.cpp**, **.py** alebo **.pas**.

PRASK
III. ročník, 2016/17
Katedra základov a vyučovania informatiky FMFI UK,
Mlynská Dolina, 842 48 Bratislava

Úlohy 1. kola zimnej časti

Termín odoslania riešení tejto série je pondelok **22. máj 2017**.

Teoretické úlohy

V tejto časti ťa čaká niekoľko matematickejších úloh, ktoré úzko súvisia s informatikou. Ako riešenie týchto úloh treba poslať podrobne spísaný postup toho, ako si riešil danú úlohu.

A ak by ťa to zaujímalo, podobné úlohy môžeš nájsť aj v Olympiáde v informatike, kategória B (http://oi.sk/archiv/2016/sl-2016-1-zad-B.pdf). Vrelo ti ju odporúčame riešiť tiež, naučíš sa veľa nových vecí a môžeš sa dostať aj na krajské kolo Olympiády.

1. Pohodový festival

15 bodov za riešenie

Ak máte akékoľvek otázky ohľadom tejto úlohy, napíšte Žabe na zaba@ksp.sk

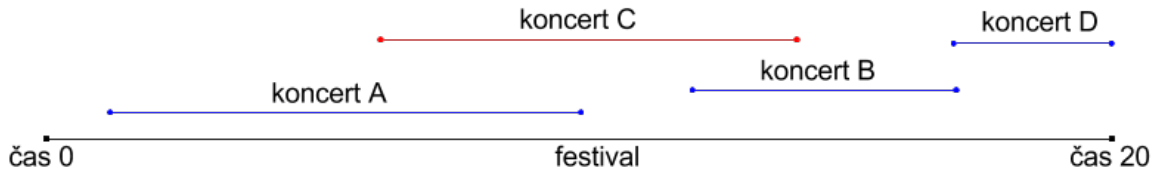
Leto sa blíži a s ním aj Sysľov obľúbený hudobný festival. Tak ako každý rok, aj teraz naň príde veľa skvelých kapiel, ktoré by si Syseľ rád vypočul. Bohužiaľ, nedá sa stihnúť všetko, pretože jednotlivé koncerty sa prekrývajú. Aby však maximalizoval svoj hudobný zážitok, chcel by ich Syseľ stihnúť čo najviac. Viete mu poradiť, ktorých koncertov sa má zúčastniť?

Úloha

Festival začína v čase 0 a uskutoční sa na ňom niekoľko koncertov. Každý koncert má pevne určený čas začiatku a čas konca. Samozrejme, každý koncert skončí neskôr ako začal a začiatok ani koniec sa nebudú posúvať[3]. Inak sa však koncerty môžu ľubovoľne prekrývať.

Keď si Syseľ vyberie niektorý koncert, chce sa ho zúčasniť celého. Nemôže teda počas neho odísť na nejaký iný koncert. Ak však nejaké vystúpenie končí v ten istý čas ako začína iné, Syseľ vie stihnúť oba koncerty. Naviac nemá žiadne preferencie, takže jeho cieľom je naozaj iba maximalizovať počet koncertov, ktorých sa zúčastní.

Predstavme si, že na festivale sú 4 koncerty – koncert `A` začína v čase 1 a končí v čase 10, koncert `B` začína v čase 12 a končí v čase 17, koncert `C` začína v čase 6 a končí v čase 14 a koncert `D` začína v čase 17 a končí v čase 20. Vidíme, že najlepšie, čo vie Syseľ spraviť je ísť na koncerty `A`, `B` a `D`. Koncert `C` sa totiž prekrýva s koncertom `A` aj `B` a ak by naň Syseľ išiel, stihol by najviac jeden ďalší koncert (`D`).



a) (3 body) Syseľ sa rozhodol vyberať si koncerty nasledovným spôsobom – ako na prvý ide na koncert, ktorý začína čo najskôr. Následne vždy keď skončí koncert, na ktorom je, ponáhľa sa na koncert s najbližším začiatokm.

V našom príklade teda ide na koncert `A`. Ten skončí v čase 10. Koncert `C` je vtedy v plnom prúde a Syseľ ho už nestíha. Najbližšie začína koncert `B` a Syseľ sa ho zúčastní. Keď v čase 17 skončí aj ten, Syseľ si všimne, že ešte stíha začiatok koncertu `D` a preto sa tam ponáhľa. Po konci koncertu `D` sa už žiaden ďalší nekoná a Syseľ ide domov.

Takýto spôsob vyberania koncertov je však **chybný**. Vytvorte taký zoznam koncertov (s ich začiatkami aj koncami), že opísaný postup na nich nenájde najlepšie možné riešenie. Na vašom príklade tiež ukážte, aké riešenie nájde popísaný postup a vysvetlite, prečo toto riešenie nie je optimálne.

[3] Ani keby diváci žiadali prídavok akokoľvek úpenlivo.

b) (4 body) Po neúspechu z podúlohy a) prišiel Syseľ s novou metódou. Najskôr si pre každý koncert vypočíta, s koľkými inými koncertami sa prekrýva. Následne si vyberie koncert s najmenším počtom prekryvov. Zo zoznamu koncertov vymaže tento vybraný koncert a aj všetky koncerty, s ktorými sa prekrýval (a ktorých sa teda nebude môcť zúčastniť). No a opísaný postup (aj s počítaním prekryvov) opakuje na zvyšných koncertoch, až kým ich všetky nevyškrtá.

V našom príklade si teda spočíta, že koncerty `A` a `B` majú 1 prekryv, koncert `D` má 0 prekryvov (koncerty `B` a `D` sa neprekrývajú) a koncert `C` má 2 prekryvy. Vyberie si preto koncert `D` a vyhodí ho zo zoznamu koncertov (a iba ten, pretože sa s ničím neprekrýva). Ostali mu tri koncerty, s počtom prekryvov 1, 1 a 2.

Keďže mu nezáleží na tom, ktorý z koncertov s 1 prekryvom si vyberie, vyberie si napríklad koncert `A`. Zo zoznamu koncertov teda musí vyhodiť koncert `A`, ale aj koncert `C`, s ktorým sa `A` prekrýval. Ostal mu už len jeden koncert `B` a toho sa teda tiež zúčastní.

Aj tento spôsob je však **chybný**. Vytvorte taký zoznam koncertov (s ich začiatkami aj koncami), že opísaný postup na nich nenájde najlepšie možné riešenie. Na vašom príklade tiež ukážte, aké riešenie nájde popísaný postup a vysvetlite, prečo toto riešenie nie je optimálne.

c) (8 bodov) Sysľa už nič ďalšie nenapadlo, preto je na vás aby ste vymysleli postup, ktorým má Syseľ vyberať koncerty tak, aby ich vybral najväčší možný počet. Vo svojom riešení popíšte algoritmus – teda postupnosť krokov a pravidiel, ktoré určia, ktorých koncertov sa má Syseľ zúčastniť. Tento postup má pritom fungovať bez ohľadu na to, kedy začínajú a končia jednotlivé koncerty a koľko týchto koncertov je. Ak existuje viacero rovnako dobrých riešení, je jedno, ktoré z nich váš postup nájde.

Dôležitou časťou vašeho riešenia musí byť aj **dôkaz správnosti**. Skúste popísať, prečo je vaše riešenie správne a prečo vždy nájde správnu odpoveď. Pokúste sa vyhnúť tvrdeniam `„Na týchto koncertoch to funguje..."` a `„V takomto prípade to bude správne..."`. Namiesto toho vysvetlite, prečo vždy, keď váš algoritmus vyberie nejaký koncert, tak tým nič nepokazí bez ohľadu na to, ako sú umiestnené zvyšné koncerty.

2. Pestovanie ovocia v Absurdistane

15 bodov za riešenie

Ak máte akékoľvek otázky ohľadom tejto úlohy, napíšte Samkovi Gurskému na ***hodobox@ksp.sk***

Genetickí inžinieri v Absurdistane urobili nedávno prevratné pokroky v genetickej modifikácii rastlín. Ich metóda je taká presná, že vedia určiť, koľko a dokonca aj akých plodov na vyšľachtenej rastline vyrastie. Napríklad vytvorili jahodu, na ktorej v lete dozrie jedna jahoda a jedna malina.

Naviac, aj nová jahoda, ktorá na rastline vyrástie, má tú istú vlastnosť. Preto ak z nej zoberú semiačko (nachádza sa tam iba jedno) a zasadia ho, v lete ďalšieho roka vyrastie rastlina s jednou jahodou a jednou malinou. Malinu zase vyšľachtili tak, aby z jej semiačka vyrástla vždy iba jedna malina (a žiadne jahody).

Označme si jahodu písmenom `J` a malinu písmenom `M`. Ak v prvom roku zasadíme jednu jahodu a následne budeme každý rok sadiť semienka zo všetkých plodov, ktoré nám dozrejú, budeme mať v nadchádzajúcich rokoch tieto plody:

- 1-vý rok – `J` – na začiatku máme jednu jahodu, ktorú zasadíme.
- 2-hý rok – `JM` – na zasadenej jahode vyrástla jedna jahoda a jedna malina. Táto rastlina následne uhynula, takže do ďalšieho roka budeme sadiť iba to, čo sme dopestovali.
- 3-tí rok – `JMM` – zo zasadenej jahody vyrástla opäť jedna jahoda a jedna malina, a zo zasadenej maliny vyrástla tiež jedna malina. Dokopy máme jednu jahodu a dve maliny, ktoré budeme sadiť ďalší rok.
- 4-tý rok – `JMMM`

Vidíme, že vďaka takto vyšľachteným rastlinám vieme dokonale predpokladať úrodu v ľubovoľnom ďalšom roku. V n-tom roku totiž budeme mať jednu jahodu a $n-1$ malín, teda dokopy presne n plodov.

To, samozrejme, zaujalo vládu Absurdistanu. Vďaka štatistikám a vešteckej guli totiž presne vie, ako sa bude rok po roku vyvíjať spotreba ovocia v ich krajine. Ak by teda dokázali vyšľachtiť rastliny, ktoré túto spotrebu dokonale pokrývajú, bol by to veľký úspech pre ich ekonomiku. A práve na tomto mieste vstupujete do príbehu vy.

Úloha

Rastliny budeme v tejto úlohe označovať veľkými písmenami anglickej abecedy. Každá rastlina bude mať naviac určené, aké plody na nej dozrievajú. Toto zapíšeme tak, že za písmeno, ktoré zodpovedá rastline, napíšeme šípku a za tú písmená tých rastlín, ktorých plody na nej dozrejú. V našom prípade by sme teda mali rastliny `J`

→ `JM` (na poradí písmen na pravej strane nezáleží, rastlina `J` → `MJ` je úplne rovnaká) a `M` → `M`. Dajte si pozor, že každá rastlina musí mať priradené **práve jedno** takéto pravidlo. Naviac, ak by sme chceli aby na nejakej rastline nič nevyrástlo (a takáto rastlina proste umrie), zapíšeme to pomocou → 0.

Na začiatok si potom určíme, ktoré rastliny máme v roku 1. Počet týchto rastlín môže byť buď určený vládou alebo si ho v niektorých prípadoch môžeme určiť sami. V tom prípade však nemôžeme začínať s **viac ako 5** rastlinami. To, akých typov sú tieto rastliny, je jedno a môžeme si to určiť ako chceme.

Následne v každom roku zasadíme všetky rastliny, ktoré máme. Na nich nám vyrastú nejaké plody, ktoré zozbierame, a pôvodné rastliny zahynú. Ďalší rok potom zasadíme všetky zozbierané plody[4], z nich nám opäť niečo vyrastie a toto opakujeme rok čo rok.

Vláda Absurdistanu vám zadá nejaké nekonečné postupnosti, ktoré hovoria o tom, koľko plodov treba vypestovať v ktorom roku. Vašou úlohou bude navrhnúť také rastliny, aby počet plodov, ktoré tieto rastliny vyprodukujú, bol rok po roku rovnaký ako v zadanej postupnosti.

Príklady

Skôr ako sa pustíme do súťažných úloh, ukážme si niekoľko užitočných príkladov.

Príklad 1: v n-tý rok chceme vypestovať n plodov. Takúto úlohu môžeme zapísať ako postupnosť $1, 2, 3, 4, \ldots$

Toto je vlastne príklad zo zadania, preto sú riešením rastliny `J` → `JM` a `M` → `M`. A začínať chceme s jednou rastlinou `J`.

Príklad 2: v n-tom roku chceme vypestovať $2n + 1$ plodov. Takúto úlohu môžeme zapísať ako $3, 5, 7, 9 \ldots$

Riešenie bude používať tie isté rastliny, ako sme použili v prvom príklade. Akurát v roku 1 musíme mať 3 rastliny. Zvolíme si rastliny `JJM`. Ešte potrebujeme dokázať, že takéto riešenie je správne. Z predchádzajúceho príkladu vieme, že za n rokov vyrastie z jednej `J` dokopy n plodov. No a `M` sa za celú dobu nezmení. V n-tý rok teda budeme mať $n + n + 1 = 2n + 1$ plodov.

Príklad 3: je nám jedno, koľko plodov vypestujeme v prvom roku, potom však budeme chcieť, aby sa počet plodov každým rokom zdvojnásobil. Takéto zadanie zapíšeme ako $x, *2, *2, \ldots$ – x na začiatku hovorí o tom, že vládu nezaujíma počet plodov v prvom roku a môžeme si tento počet určiť sami (akurát musí byť najviac 5).

Toto môžeme dosiahnuť napríklad pomocou rastliny `X` → `XX`, pričom na začiatku budeme začínať s jednou takotou rastlinou. Vidíme, že jediné rastliny, ktoré môžeme vypestovať, sú rastliny `X`. No a každý rok nám z každého zasadeného `X` vyrastú dva plody `X`, preto sa bude ich počet zdvojnásobovať.

Úloha

Pre každú z nasledovných podúloh navrhnite sadu rastlín a typy rastlín v prvom roku tak, aby bol počet plodov v roku n taký, ako určila vláda Absurdistanu vo svojej predpovedi. Je jasné, že vaše riešenie nemôže obsahovať viac ako 26 rôznych rastlín pre jednu podúlohu (lebo ako by sme ich pomenovali?). Vzorové riešenie však nepotrebuje na žiadnu podúlohu viac ako 10 rastlín.

Vo svojom riešení tiež **zdôvodnite**, prečo daný návrh funguje a v n-tom roku vyprodukuje požadovaný počet plodov. Snažte sa argumentovať všeobecne, pokúste sa vyhnúť argumentom `„Pre prvé štyri roky to funguje..."`.

a) (1 bod) $1, 2, 3, 1, 2, 3, 1, \ldots$

b) (2 body) $x, +2, -1, +2, -1, +2, \ldots$

c) (2 body) $x, *4, /2, *4, /2, \ldots$

d) (3 body) $x, *2, +1, *2, +1, *2, \ldots$

e) (3 body) n-té fibonacciho číslo: $1, 2, 3, 5, 8, 13, \ldots$. n-té fibonacciho číslo vypočítate ako súčet dvoch predchádzajúcich čísel. Nasledujúce číslo je teda číslo $8 + 13 = 21$.

f) (4 bodov) n^2: $1, 4, 9, 16, 25, \ldots$

Praktická úloha

Pri práci s počítačom je potrebné vedieť pracovať aj s rôznymi nástrojmi, ktoré slúžia na úpravu obrázkov, prácu so zvukom či vyhľadávaním na internete. V tejto časti ťa preto zakaždým čaká nejaká netradičná úloha.

[4] V skutočnosti nesadíme plody, ale semiačka z nich. Plody zjedia občania Absurdistanu a podľa zákona z každého zjedeného plodu musia štátu odovzdať práve jedno semiačko. A keďže štatistiky sú dokonalé, zkonzumuje sa každý plod.

3. Pofiderné kasíno

15 bodov za riešenie

Ak máte akékoľvek otázky ohľadom tejto úlohy, napíšte Prefixovi na michalsladecek98@gmail.com

Adam má veľmi rád rezne. Preto keď zistil, že si ich v školskej jedálni môže objednať koľko len chce, bol veľmi nadšený. Rovno si ich preto objednal 100 000 a tešil sa, ako ich bude všetky jesť.

Na druhý deň sa cítil ako v raji. Rezeň sem, rezeň tam... Skôr ako ich všetky zjedol si však uvedomil, že za ne bude musieť aj zaplatiť. A keďže toľko peňazí pri sebe nemá, ocitol sa v obrovských dlhoch. A kuchárky s tĺčikmi na mäso mu prišli oznámiť, že ak nezaplatí do konca PRASK kola, vyklepú ho viac, ako tých 100 000 rezňov.

Adam preto potrebuje rýchlo zarobiť veľké peniaze. A kde inde ako v kasíne[5]! Samozrejme, Adam vie, že v kasíne sa ho budú snažiť oklamať. On je však šikovnejší. A naviac má kamaráta, ktorý programoval stroje v kasíne a ten mu poskytol ich zdrojové kódy. Teraz už len zistiť, ako vyhrať. Pomôžete mu?

Úloha

V každej úlohe dostanete zdrojový kód jedného výherného stroja. Vašou úlohou bude nájsť v tomto programe slabinu a potom si ísť zahrať do nášeho kasína. Vždy keď sa vám na nejakom automate podarí vyhrať, dostanete url adresu, na ktorej sa vám pripočítajú body. A ak prehráte, nič sa nedeje, môžete to skúsiť znovu.

Na hranie v kasíne sa potrebujete pripojiť na náš server, kde toto kasíno beží. Návod ako to spraviť pre rôzne operačné systémy je tu.

Linux:

Treba si nainštalovať netcat – je možné, že ho už máte. Ak nie, tak si ho v závislosti od distribúcie, ktorú používate nainštalujte.

Pre Ubuntu by napríklad fungoval príkaz:

```
sudo apt-get install netcat
```

Následne stačí do konzoly zadať príkaz:

```
nc 158.195.16.154 9947
```

ktorý vás pripojí na náš server a vy môžete hrať.

Windows:

K tomuto tutoriálu existuje aj videonávod.

Netcat si najskôr musíte stiahnuť. Následne si v nejakom priečinku tento `.zip` rozbaľte. Potom si otvoríte príkazový riadok (buď zo `Start` menu alebo cez spúšťač programov – stlačíte `Windows + R` a napíšete `cmd`). V ňom musíte prejsť do priečinku s rozbaleným netcat zipom. To spravíte príkazom:

```
cd cesta_k_netcatu
```

pričom hodnotu `cesta_k_netcatu` zistíte tak, že v prieskumníku súborov sa pozriete na adresu priečinka s týmto zipom.

Následne zadáte príkaz:

```
chcp 65001
```

ktorým nastavíte vypisovanie slovenských znakov v príkazovom riadku. Ostáva už len pripojiť sa na náš server príkazom:

```
nc.exe 158.195.16.154 9947
```

V kasíne používame **Python, verziu 3.5.2.**. Ak si teda chcete skúšať spúšťať programy uvedené nižšie u seba, použite túto verziu.

Ak budete mať akékoľvek **otázky ohľadom úlohy**, toho ako funguje netcat, Python, C++ alebo naše generátory náhodných čísel, **napíšte Prefixovi**. Rád vám pomôže.

Podúlohy

[5] Toto doma neskúšajte.

a) (2 body)

Adam vie, že prístroj používa veľmi jednoduchú funkciu na generovanie náhodných čísel:

```
def random():
    global seed1
    seed1 = (a*seed1)%10000
    return seed1
```

V tomto kóde je a náhodné číslo, ktoré je stanovené pri spustení kasína a počas celého behu sa nemení (zmení sa však ak napr. reštartujete kasíno). Adam toto číslo nepozná. Číslo $seed1$ je globálna premenná, ktorá má na začiatku tiež neznámu hodnotu. Vždy keď Adam spraví ďalší pokus pomocou hodnoty a a $seed1$ sa vypočíta nová hodnota pre $seed1$, ktorá slúži ako výsledok. Ak je tento výsledok rovnaký ako Adamov pokus, Adam vyhráva. Znak „`%`" predstavuje zvyšok po delení (modulo).

b) (3 body)

Zákerný programátor tohto automatu sa chcel uistiť, že súťažiaci v žiadnom prípade nemôže vyhrať. Do už existujúceho programu preto pridal jeden riadok, ktorý mal túto podmienku zaručiť. Program bol však písaný v C++ a zlý programátor sa dopustil osudnej chyby. Zistíte akej? Ako môžete vyhrať?

```
//Hodnotu a nepoznáte, medzi pokusmi sa však nemení
int a;

bool vyhral(){
    int x,y;
    cout << "Ake x tipujes?";
    cin >> x;               // načíta prvé číslo
    cout << "Ake y tipujes?";
    cin >> y;               // načíta druhé číslo
    int z = x*a + y;
    if(z<0) z = -z;         // pridaný riadok
    if(z<0) return true;    // Výhra - nemožné?
    else return false;      // Prehra
}
```

Hint: Čo je to vlastne ten `int` v C++? A ako presne funguje?

c) (4 body)

Po fiasku s prístrojom z úlohy a) sa kasíno rozhodlo, že obmení svoju náhodnú funkciu. Nová funkcia vyzerá takto:

```
def random():
    global seed1
    seed1 = (a*seed1 + b)%9973
    return seed1
```

Adam v tomto prípade nepozná dokonca dve čísla – a a b.

d) (6 bodov)

Koniec hier. Použitie funkcie `randint` z Pythonu sa už určite nebude dať nijak oklamať:

```
seed(int(time.time()))

def random():
    return randint(0,1000000)
```

Tento program funguje tak, že na začiatku sa generátor náhodných čísel nastaví podľa aktuálneho času (prvý riadok). Je to vlastne nastavenie hodnoty *seed*1 z podúloh a) a b). Následne sa podľa tohto nastavenia postupne generuje nové číslo, ktoré je použité na generovanie toho ďalšieho...

Adam našťastie pozná približný čas, v ktorom bol tento generátor spustený (a dozviete sa ho aj vy, keď si spustíte kasíno). Nie je to však také ľahké, lebo pred vami už na ňom nejaký ľudia hrali a preto aktuálna hodnota môže byť dosť posunutá. Našťastie viete, že nebola zavolaná viac ako 1 000 krát.

Hint: unixový čas

Programátorské úlohy

Tieto úlohy sú zamerané na praktickú tvorbu programov v niektorom vyššom programovacom jazyku ako je napríklad Python, C++ alebo Pascal. Na stránke odovdzávaš **iba zdrojový kód** svojho programu riešiaceho zadanú úlohu, ktorý bude okamžite automaticky otestovaný a do pár sekúnd sa dozvieš, koľko bodov tvoj program získal. Tieto body ti už nikto nemôže zobrať, ale ak si nezískal plný počet bodov, môžeš opakovane odovzdávať opravený program, až kým nebudeš spokojný s výsledkom.

Ak už vieš programovať, ale ešte si nepracoval s naším testovacím systémom, odporúčam ti zájsť na Programátorskú Liaheň (`http://betaliahen.ksp.sk`), kde si o tom môžeš prečítať úvodný text a vyriešiť si niekoľko jednoduchých úloh.

Ak však **nevieš programovať, tak nezúfaj!** Pripravili sme pre teba **Programátorskú Liaheň**, ktorá ťa **naučí základy programovania** v jazyku C++. Navyše, za riešenie týchto tutoriálových úloh na Liahni môžeš získať body priamo do PRASKu a tým si vynahradiť neriešenie niektorej z programátorských úloh.

Presnejšie to funguje takto. Na Liahni sa nachádzajú dve sady úloh, prvá zameraná na premenné a druhá na podmienky v jazyku C++. V týchto sadách sa nachádzajú bodované aj nebodované úlohy, ktoré môžeš postupne riešiť a ktoré ti postupne vysvetlia danú problematiku. Dokopy sa v jednej sade dá získať až 15 bodov.

Týmito bodmi si potom môžeš nahradiť úlohy 4 a 5. Samozrejme, toto môžeš urobiť **s každou sadou najviac raz**.

No a v budúcej sérii pribudnú na Liahni ďalšie dve sady, ktorými si budeš môcť opäť nahradiť programátorské úlohy.

Samozrejme, nič ti nebráni riešiť aj úlohy z Liahne aj klasické programátorské úlohy v PRASKu.

Programátorskú Liaheň nájdeš na tejto stránke: `http://betaliahen.ksp.sk`

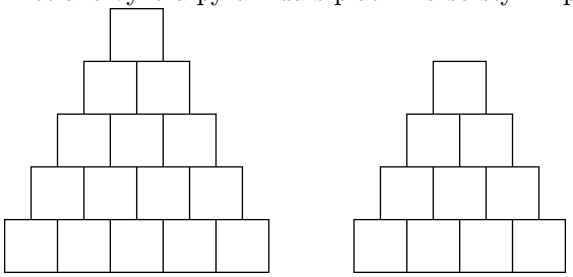
4. Pyramídy

15 bodov za riešenie

*Táto úloha sa dá nahradiť riešením sady **variables_cpp** na Liahni (betaliahen.ksp.sk) . Ak chceš, aby ti namiesto bodov za riešenie tejto úlohy boli započítané body získané riešením spomínanej sady, na stránke odovdzaj pdf-ko s prezývkou, ktorú používaš na Liahni.*

Ak máte akékoľvek otázky ohľadom tejto úlohy, napíšte Baške na `baska@ksp.sk`

Malý Miško dostal na narodeniny úplne novú sadu drevených kociek. Jeho obľúbenou stavbou je pyramída[6]. Na obrázku nižšie si môžete pozrieť ako vyzerá pyramída s piatimi a so štyrmi poschodiami.



Spodný riadok k-poschodovej pyramídy sa skladá z k kociek. Každé ďalšie poschodie má o jednu kocku menej ako to pod ním. Takže počet kociek v pyramíde vieme vypočítať ako súčet čísel od 1 (najvrchnejšie poschodie) po k (najspodnejšie poschodie).

V matematickej knižke svojho otca dokonca Miško zistil, že tento súčet sa dá vyrátať aj pomocou jednoduchého vzorca:

$$1 + 2 + \ldots + k = \frac{k \cdot (k+1)}{2}$$

[6] Miško zastáva názor, že obyčajné veže sú príliš nestabilné a môžu sa ľahko zrútiť.

Aby urobil rodičom radosť, chcel by Miško postaviť jednu pyramídu pre maminku a jednu pre otecka. Zaujíma ho preto, či vie postaviť **práve dve** pyramídy a pritom použiť **všetky** kocky, ktoré má.

A keďže ho rodičia ešte nenaučili programovať[7], poprosil o pomoc vás.

Úloha

Na vstupe je číslo n – počet Miškových kociek.

Zistite, či vie postaviť dve (nie nutne rovnako vysoké) pyramídy, ktoré dokopy obsahujú práve n kociek. Samozrejme, pyramída, ktorá má 0 poschodí, sa za pyramídu nepovažuje.

Vstup

Vstup obsahuje jediný riadok s kladným celým číslom n.

V sadách 4 a 5 bude číslo n naozaj veľké a nezmestí sa vám do štandardných 32 bitových premenných. Namiesto nich môžete použiť **64 bitové premenné** (`long long` v C++, `Int64` v Pascale). Takto veľké premenné odporúčame používať aj na ukladanie medzivýsledkov a pomocných premenných. V prípade, že používate Python tento problém nenastane, pretože premenné v Pythone môžu byť ľubovoľne veľké.

Výstup

Ak sa zo všetkých Miškových kociek dajú postaviť dve pyramídy, vypíšte slovo `„ANO"`. V opačnom prípade vypíšte slovo `„NIE"`.

Odpoveď vypisujte bez úvodzoviek okolo slov, všetko veľkými písmenami a nezabudnite vypísať koniec riadku!

Hodnotenie

Váš program bude spustený na piatich sadách vstupných súborov. Body dostanete za každú úspešne vyriešenú sadu. Obmedzenia na veľkosť čísla n v jednotlivých sadách nájdete v nasledujúcej tabuľke.

Číslo sady	1	2	3	4	5
maximálne n	100	10 000	1 000 000	10^{10}	10^{12}

Príklady

vstup	výstup
`256`	`ANO`

Miško môže postaviť 2-poschodovú pyramídu z 3 kociek a 22-poschodovú pyramídu z $\frac{22 \cdot 23}{2} = 253$ kociek.

vstup	výstup
`512`	`NIE`

Z 512 kociek sa mu nepodarí postaviť dve kompletné pyramídy.

5. Poriadna výzva

15 bodov za riešenie

*Táto úloha sa dá nahradiť riešením sady **conditions_cpp** na Liahni (betaliahen.ksp.sk) . Ak chceš, aby ti namiesto bodov za riešenie tejto úlohy boli započítané body získané riešením spomínanej sady, na stránke odovzdaj pdf-ko s prezývkou, ktorú používaš na Liahni.*

Ak máte akékoľvek otázky ohľadom tejto úlohy, alebo máte napríklad problémy s načítaním vstupu, napíšte Andrejovi na andrejkorman1@gmail.com

Žaba sa po sústredení opäť zastavil doma v Lučenci. Pri obede sa pochválil ako bolo na sústredení super a čo všetko sa deti naučili. Všetci pri stole boli ohromení.

Jedine babka ostala skeptická a pochybovačne odvetila: “Algoritmy, polia, časová zložitosť... Zaujímavé to veci, ale povedz mi Miško, dokážu sa tie vaše decká popasovať aj s nejakým skutočným problémom?”

Žabovi sa začali potiť dlane a potichu uvažoval, na čo môže babka narážať. Ak by chcela šľachtiť kvetiny, tak decká naučil algoritmus na porovnávanie DNA, takže by to malo byť v pohode. A keby chcela robiť vzory na vyšívanie, tak na to spravila jeho spolužiačka ako bakalársku prácu aplikáciu[8]. Ale ak to bude niečo iné...

[7] Ale už čoskoro...

[8] True story.

Pri stole nastalo hrobové ticho. Babke po chvíli premýšľania skĺzli oči na noviny a v nich vidí poslednú nevyriešenú osemsmerovku. V tom jej zablysnú oči.

"A povedz mi Miško, dokázali by tie tvoje deti vyriešiť moje osemsmerovky? S niektorými si ozaj neviem dať rady," pýta sa babka.

Žaba neváhal a prijal babkinu výzvu.

Pre tých, ktorí nie sú úplne oboznámení s touto hrou: Ide o hru, kde dostanete mriežku písmen a niekoľko slov. V mriežke písmen sa snažíte nájsť a vyškrtnúť zadané slová. Ako už názov napovedá, slová sa môžu v mriežke nachádzať vo ôsmich základných smeroch. Po vyškrtaní všetkých slov ostanú niektoré písmená nepreškrtnuté. Keď si ich zapíšete v poradí odhora dole a zľava doprava, dostanete riešenie osemsmerovky. Zväčša je to odpoveď k nejakej hádanke alebo vtipu.

Úloha

Na vstupe dostanete osemsmerovku a zoznam slov, ktoré v nej máte nájsť. Vašou úlohou je nájsť riešenie osemsmerovky, teda písmená, ktoré ostanú po vyškrtaní všetkých týchto slov.

Slovo v osemsmerovke je postupnosť písmen v jednom z ôsmich smerov. Je zaručené, že žiadne slovo sa v osemsmerovke nenachádza viackrát. Slová sa môžu v osemsmerovke prekrývať, avšak platí, že žiadne nie je úplne prekryté iným slovom.

Vstup

Na prvom riadku vstupu dostanete tri čísla: r, s a n. Číslo r určuje počet riadkov a číslo s počet stĺpcov osmesmerovky. n je počet slov, ktoré je v osemsmerovke treba vyškrtnúť.

Na každom z ďalších r riadkov je s znakov, ktoré popisujú samotnú osemsmerovku. Nakoniec je na vstupe n slov, každé na samostatnom riadku.

Znaky na vstupe predstavujúce osemsmerovku a hľadané slová sú malé písmená anglickej abecedy.

Pri riešení v jazyku C++ vám odporúčame využiť dátovú štruktúru `string`. K jej použitiu je potrebné na začiatok vášho programu pridať `#include<string>`. Do `string`u potom viete pomocou `cin` >> načítavať vstup a pracovať s ním ako s poľom, v ktorom je na každej pozícii jeden znak (typ `char`).

Opäť pripomíname, že ak by ste mali problémy s načítavaním a spracovaním vstupu, či už v C++, Pythone alebo niečom inom, napíšte Andrejovi.

Výstup

Na výstup vypíšte písmená, ktoré ostali po vyškrtaní všetkých slov, v poradí, v akom by ste ich čítali pri klasickej osemsmerovke. Výstup ukončite znakom konca riadku.

Hodnotenie

Vaše riešenie bude hodnotené na viacerých sadách testovacích vstupov. V preklade to znamená, že aj pomalšie korektné riešenia dostanú nejaký počet bodov a teda sa ich oplatí odovzdať.

Vo všetkých sadách bude platiť, že $r, s \leq 100$ a $n \leq 40$. Zároveň platí, že celkový súčet dĺžok slov neprekročí 400.

V jednotlivých sadách platia nasledujúce obmedzenia:

- V prvej sade platí, že $r = 1$ a $n = 1$. Osemsmerovka má teda iba jeden riadok a hľadáte v nej iba jedno slovo.
- V druhej sade platí, že $r = 1$. Osemsmerovka má iba jeden riadok, hľadáte v nej však viacero slov.
- V tretej sade platí, že $n = 1$. To znamená, že hoci osemsmerovka môže byť ľubovoľne veľká, vy v nej budete hľadať iba jediné slovo.
- V štvrtej sade platí, že $r \cdot s \leq 50$. Teda osemsmerovka bude relatívne menšia.
- Pre piatu sadu neplatia žiadne špeciálne obmedzenia.

Za vyriešenie každej z týchto sád získate 3 body.

Príklady

vstup

```
9 9 15
zapamkacg
kkjadrorn
ioyrylati
askznmaes
stssaiyhu
erttmjpsl
railistrk
dkonkalvy
aaskatsec
adresa
cesta
cyklus
dynamika
gramatika
halda
jadro
jazyk
kostra
list
mapa
mys
skok
tok
vlakno
```

výstup

```
zacniriesitprask
```

Táto osemsmerovka sa nachádzala na letáku PRASKu

vstup

```
5 4 3
ccxb
xabf
xaxf
xaxf
xaxf
abb
aaaa
cc
```

výstup

```
xxfxxfxxfxxf
```