Leták zimnej časti II. 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
II. ročník, 2015/16
Katedra základov a vyučovania informatiky FMFI UK,
Mlynská Dolina, 842 48 Bratislava

Úlohy 2. kola zimnej časti

Všetci vedúci PRASKu Vám želajú veľa šťastia a zdravia v roku 2016. A hlavne veľa dobrých riešení a rastúci záujem o informatiku :)

Termín odoslania riešení tejto série je pondelok **2. februára 2016**.

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.

V úlohe číslo 2 ťa čaká zaujímavý model – regulárne výrazy, ktorý má veľké uplatnenie aj v praktickom živote. Naviac, rovnaký model (s podobnými úlohami) sa nachádza aj v tohtoročných zadaniach Olympiády v informatike, kategória B.

1. Preblšený cirkus

15 bodov za riešenie

Ak máte akékoľvek otázky ohľadom tejto úlohy napíšte Maji Vajdovej na maja@ksp.sk

V jednom malom neznámom cirkuse na okraji jedného malého bezvýznamneho mesta sa jedného nudného dňa premnožili blchy. Celý cirkus sa obrátil hore nohami. Namiesto toho, aby sa nacvičovali nové čísla, zvieratá aj ľudia začali splašene pobehovať po okolí a pomaly cirkus opúšťať. Keď to už vyzeralo tak, že malý neznámy cirkus bude treba zavrieť, blchy zmutovali do obrovských rozmerov.

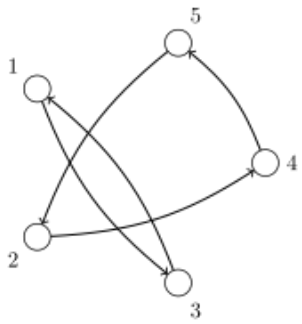
To, čo sa na prvý pohľad javilo ako ešte väčšia katastrofa, sa nakoniec ukázalo ako výhoda, ktorá náš malý cirkus preslávila po celom svete. Také veľké blchy sa totiž dajú trénovať, preto sa náš cirkus premenoval na blší a funguje dodnes.

Pred blším vystúpením sa vždy na podlahu nakreslí niekoľko kruhov a očíslujú sa začínajúc od 1. Do každého sa potom položí jedna blcha. Vždy, keď počas vystúpenia cvičiteľ zapíska na píšťaľku, všetky blchy skočia zo svojho kruhu do nejakého (nie nutne iného) určeného kruhu. Kruh, do ktorého má blcha skočiť je určený číslom kruhu, v ktorom blcha stojí. Napríklad môže byť povedané, že z kruhu 3 sa skáče do kruhu 1 a z kruhu 2 sa skáče späť do kruhu 2. Toto určenie sa počas celého vystúpenia nemení.

Cvičitelia by však potrebovali pomôcť s plánovaním nových choreografií a s prosbou o pomoc sa obrátili na vás.

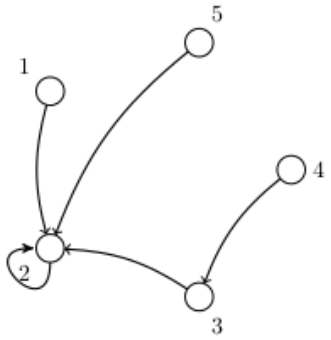
Úloha

Máme choreografiu pozostávajúcu z n kruhov pre blchy a z n bĺch. Pre každý kruh je určené, do ktorého kruhu sa z neho blcha musí pri zapískaní pohnúť. Na obrázku je vidno rozostavenie pre 5 bĺch. Šípka vedúca z kruhu určuje, do ktorého kruhu má z neho blcha skočiť.



Na ďalšom obrázku je vidno inú choreografiu pre 5 bĺch. Môžete si na nej všimnúť, že po dvoch zapískaniach sa všetky blchy ocitnú v kruhu 2. Takéto blšie choreografie voláme *stretávacie*. Aj blšia choreografia na

predchádzajúcom obrázku má zaujímavú vlastnosť: po presne šiestich zapískaniach bude každá blška naspäť v krúžku, v ktorom bola na začiatku vystúpenia. Takéto blšie choreografie voláme *opakovacie*.



Vašou úlohou je vymyslieť niekoľko choreografií tak, aby spĺňali nasledovné požiadavky. Vo vašom riešení však nestačí nakresliť žiadanú choreografiu, ale aj vysvetliť, prečo má daná choreografia požadované vlastnosti. Choreografie bez popisu budú hodnotené najviac polovicou bodov.

a) (1 bod) Nakreslite opakovaciu choreografiu, pre ktorú sa blšky na pôvodné miesta vrátia nie po 6, ale po 7 skokoch. Môžete použiť ľubovoľný počet krúžkov, nemusí ich byť len 5.

b) (1 bod) Nakreslite opakovaciu choreografiu, v ktorej sa blšky na pôvodné miesta vrátia prvýkrát až po 10 skokoch. Tentokrát ale musíte použiť menej ako 10 krúžkov.

c) (2 body) Choreograf by od vás chcel blšiu choreografiu s 10 krúžkami. Zaujímalo by ho, aké je najväčšie číslo k, pre ktoré platí, že sa blšky prvýkrát ocitnú všetky naraz vo svojich počiatočných pozíciach po práve k zapískaniach. Ako taká choreografia vyzerá? Prečo ja vami nájdené k najväčšie možné?

d) (2 body) Nájdite choroegrafiu s čo najmenej krúžkami, ktorá sa prvýkrát opakuje po 3960 zapískaniach.

e) (4 body) Nájdite najmenší počet krúžkov pre choreografiu, ktorá sa má prvýkrát opakovať po n zapískaniach. Vo vašom riešení popíšte postup ako takúto choreografiu nakresliť a vysvetlite, prečo sa neoplatí kresliť túto choreografiu inak. Dbajte na správnosť vašich tvrdení a argumentov.

f) (1 bod) Skúste vymyslieť choreografiu, ktorá nebude ani stretávacia, ani opakovacia.

g) (1 bod) Janko si nakreslil choreografiu so 47 krúžkami. Neukázal ju Marienke, len jej povedal, že jeho choreografia je stretávacia. Na to Marienka povedala: "Nemusíš mi ju ukazovať. Napriek tomu som si istá, že keď b-krát zapískam, budú už všetky blšky v tom istom krúžku." Pre aké najmenšie b má Marienka pravdu bez ohľadu na to, ako vyzerá Jankova choreografia? Prečo?

h) (3 body) V stretávacej choreografii z príkladu si môžeme všimnúť šipku vedúcu z krúžku 2 naspäť do krúžku 2. Takúto šipku voláme slučka. Existuje stretávacia choreografia, v ktorej nie sú žiadne slučky? Svoju odpoveď zdôvodnite.

2. Praktické regulárne výrazy 2

15 bodov za riešenie

Ak máte akékoľvek otázky ohľadom tejto úlohy napíšte Michalovi "Žabovi" Anderlemu na zaba@ksp.sk

V minulej sérii sme sa prvýkrát stretli s regulárnymi výrazmi v tejto úlohe[3]. Opäť sa k nim však vrátime. Ako sme videli, pomocou regulárnych výrazov sa dajú hľadať alebo akceptovať rôzne zaujímavé reťazce, ktoré spĺňajú isté vlastnosti. Napríklad môžete ľahko napísať regulárny výraz, ktorý vám overí, či je zadaný reťazec e-mailová adresa. Niekedy však potrebujeme overovať aj zložitejšie vlastnosti. Napríklad deliteľnosť.

A práve o tom bude celá táto séria. Budeme sa snažiť akceptovať reťazce cifier 0 až 9, ktoré reprezentujú čísla a naviac musia byť deliteľné nejakým konkrétnym číslom. Deliteľnosť niektorými číslami sa overuje ľahko, napr. 2, 4, 5, 8 alebo 10. Inými je to o niečo zložitejšie. V rámci tohto zadania postupne vytvoríme regulárny výraz akceptujúce čísla deliteľné troma. Nemusíte sa však báť, pôjdeme na to krok po kroku. Začneme tým, že si pripomenieme ako také regulárne výrazy vytvárame.

[3]Odporúčam si prečítať vzorové riešenie, môže vám to pomôcť pri tejto úlohe.

Základy použitia regulárnych výrazov

Pre jednoduchosť sa dohodneme, že všetky objekty, ktoré budeme hľadať, budú *reťazce* tvorené obmedzenou množinou znakov. Povolené znaky budú len písmená anglickej abecedy a cifry (`„a"-„z"`, `„A"-„Z"`, `„0"-„9"`).[4]

Vyhľadávať budeme tak, že napíšeme regulárny výraz – teda nejakú *vzorku* (pattern). Vzorka popisuje ako vyzerajú reťazce, ktoré nás zaujímajú. Presnejšie, o každom reťazci vieme povedať, či vzorke zodpovedá (matches the pattern) a naším cieľom pri vyhľadávaní bude napísať vzorku tak, aby jej zodpovedali práve tie reťazce, ktoré chceme nájsť a žiadne iné. Ako uvidíme nižšie, vo vzorke sa budú môcť vyskytovať aj niektoré iné znaky ako v reťazcoch.

Teraz si popíšeme základy tvorby vzoriek – teda povieme si, z čoho sa taká vzorka môže skladať a ktoré reťazce jej potom zodpovedajú.

- Základným stavebným kameňom vzoriek sú samotné znaky z vyššie uvedeného zoznamu. Vzorke tvorenej jediným znakom zodpovedá reťazec tvorený dotyčným znakom a nič iné.
- Základná operácia so vzorkami je zreťazenie. Keď napíšeme dve vzorky za sebou, dostaneme novú vzorku. Tej zodpovedajú reťazce zložené z dvoch častí, pričom prvá zodpovedá prvej vzorke a druhá druhej. Zreťaziť samozrejme môžeme ľubovoľne veľa vzoriek. Napr. vzorke `„jablko"` zodpovedá len reťazec `„jablko"`.
- Vzorky môžeme uzatvárať do obyčajných zátvoriek. Napr. vzorke `„(jablko)"` zodpovedá len reťazec `„jablko"` a nič iné.
- Logický or (alebo) je označovaný symbolom `„|"`. Ak ním spojíme dve vzorky, dostaneme novú, ktorej zodpovedajú aj reťazce zodpovedajúce prvej, aj reťazce zodpovedajúce druhej vzorke. Napr. vzorke `„jabl(k|ck)o"` zodpovedajú reťazce `„jablko"` a `„jablcko"`.

  Zreťazenie má vyššiu prioritu ako or. Napr. vzorke `„aaa|bbb"` zodpovedajú len reťazce `„aaa"` a `„bbb"`.
- Inú možnosť, ako dať vo vzorke na výber predstavuje množina znakov, ktoré sa na danom mieste môžu vyskytnúť. Tú zapisujeme `„[znaky]"`. Napríklad vzorke `„jablk[oa]"` zodpovedajú reťazce `„jablko"` a `„jablka"`.

  Množina všetkých dostupných znakov sa skrátene zapisuje `„."`. Teda vzorke `„j....a"` zodpovedá každý 6-znakový reťazec začínajúci na `„j"` a končiaci na `„a"`.

  Zložitejšie množiny znakov môžeme zapísať pomocou rozsahov, napr. `„[a-z0-9]"` je ľubovoľné malé písmeno alebo číslica. Ak popis množiny znakov začína znakom `„^"`, znamená to negáciu: danej množine zodpovedajú všetky znaky okrem vymenovaných. Napríklad vzorke `„[a-z][^a-zA-Z]"` zodpovedajú všetky dvojznakové reťazce, ktorých prvý znak je malé písmeno a druhý znak nie je ani malé, ani veľké písmeno.
- Ak chceme, aby sa mohla časť vzorky v reťazci zopakovať, použijeme *kvantifikátor*. Základné kvantifikátory sú `„?"` (nulakrát alebo jedenkrát) a `„*"` (ľubovoľne veľakrát, aj nulakrát).

  Napríklad vzorka `„jablc?ko"` je ekvivalentná so vzorkou `„jabl(k|ck)o"`. Vzorke `„pe*s"` zodpovedajú okrem iného reťazce `„ps"`, `„pes"`, `„pees"`... Vzorke `„(ab)*c"` **nezodpovedá** reťazec `„aabbc"`, len reťazce `„c"`, `„abc"`, `„ababc"`, atď.

  Pri opakovaní vzorky jej nemusí zakaždým zodpovedať ten istý reťazec. Napríklad vzorke `„j.*a"` zodpovedá nielen reťazec `„jxxxa"`, ale aj reťazec `„jahoda"`.

  Vzorke `„<[^>]*>"` zodpovedá každý XML tag – reťazec začína znakom `„<"`, za ním nasleduje ľubovoľne veľa znakov iných od `„>"` a za nimi nasleduje znak `„>"`.

Súťažná úloha

Táto úloha má niekoľko podúloh. V každej podúlohe budete mať slovne popísanú nejakú skupinu reťazcov. Vaším cieľom bude napísať vzorku, ktorej zodpovedajú všetky popísané reťazce a žiadne iné!

a) (1 bod) Začneme zľahka. Napíšte regulárny výraz, ktorý akceptuje čísla **deliteľné piatimi**, ktoré nezačínajú zbytočnými nulami. To znamená, že má akceptovať reťazce `„0"`, `„15"`, ale nemá akceptovať reťazce `„015"` alebo `„11111"`.

[4] V skutočných regulárnych výrazoch môžeme samozrejme používať aj všetky ostatné znaky vrátane medzier. Našim obmedzením sa však vyhneme vysvetľovaniu mnohých technických detailov.

V ďalších podúlohách budeme riešiť problémy zlodejov z filmu Sám doma. Stoja na začiatku vysokých schodov, dá sa povedať že nekonečných, a chcú vyjsť hore. Každým krokom môžu buď výjsť o schod vyššie (1) alebo jeden schod prekročia, čím sa dostanú o dva schody vyššie (2). Ich cesta hore schodmi sa teda dá zapísať ako reťazec z „1" a „2". Nemusia dokonca výjsť ani úplne navrch, je možné, že na niektorom schode zastanú a zostanú tam proste stáť.

Ak by sme chceli napísať regulárny výraz, ktorý akceptuje všetky reťazce, ktoré zodpovedajú ich cestám hore schodmi, vyzeral by takto: „[12]*". Kevin však na nich prichystal pascu a niektoré schody natrel lepidlom. Tým sa chcú zlodeji väčšinou vyhnúť, lebo ak na ne stúpia, prilepia sa a ostanú tam stáť.

b) (2 body) Kevin potrel lepidlom tretí a šiesty schod. Napíšte regulárny výraz, ktorý akceptuje všetky cesty po schodoch, pri ktorých sa zlodej nenalepí na žiadny schod a vyjde aspoň na siedmy schod. To znamená, že treba akceptovať reťazce „1121222" alebo „11212", ale nemôžete akceptovať napríklad „111", „22" alebo „2112".

c) (3 body) Tentokrát sú lepidlom natreté všetky schody, ktorých číslo je deliteľné tromi. Vytvorte regulárny výraz, ktorý akceptuje všetky cesty hore schodmi, pri ktorých sa zlodej nikdy neprilepí. To znamená, že reťazec „11222212" nemáte akceptovať, lebo zlodej sa prilepí na schode číslo 6 (po tom, čo urobí „1122"). Dobré reťazce sú napríklad „11" alebo „1121212".

d) (2 body) Lepidlom sú stále natreté schody deliteľné tromi. Lepidlo je však neviditeľné a zlodeji o ňom nevedia. Preto sa počas toho, ako išli hore nalepili na jednom schode. Spravte regulárny výraz, ktorý popisuje práve tie cesty hore schodmi, ktoré skončia na schode deliteľnom troma. Správne reťazce sú napríklad „112122" alebo „111", ale nie „2212", lebo pri ňom sa zlodej nenalepil a ani „22111122", lebo zlodej sa nalepil už na schode šesť, po tom čo prešiel „2211" a teda nemohol pokračovať.

e) (1 bod) Zadanie je rovnaké ako v podúlohe d), akurát zlodej má na sebe viacero vrstiev ponožiek a keď sa nalepí na nejaký schod, najvrchnejšiu vrstvu si môže vyzuť a pokračovať v ceste. Skončiť ale musí aj tak na schode deliteľnom tromi. To znamená, že reťazec „22111122" je správny, v poriadku ale nie je reťazec „12212221", lebo takáto cesta neskončí na zalepidlovanom schode.

Odteraz to už bude veľmi podobné podúlohe e). Budeme chcieť také cesty hore schodmi, ktoré skončia na schode deliteľnom tromi, pričom medzitým sa môže prilepiť koľkokrát chce. Uvedomme si, že to zodpovedá číslam skladajúcim sa z cifier „1" a „2", ktorých ciferný súčet je deliteľný troma.

f) (1 bod) Hľadáme regulárny výraz akceptujúci čísla s ciferným súčtom deliteľným troma, ktoré môžu naviac obsahovať aj cifru „0". Napríklad teda reťazec „01200201".

g) (3 body) A pokúsme sa pridať aj zvyšné cifry. Hľadáme čísla s ciferným súčtom deliteľným trojkou, ktoré môžu obsahovať ľubovoľnú cifru „0" až „9".

h) (2 body) Regulárny výraz z podúlohy g) je takmer presne to čo hľadáme – čísla deliteľné tromi. Lebo práve tie majú ciferný súčet deliteľný tromi. Jediné, čo je naviac je, že dovoľujeme, aby číslo začínalo prebytočnými nulami. Pokúste sa opraviť tento nedostatok a vytvorte regulárny výraz, ktorý naozaj akceptuje iba čísla deliteľné troma.

V riešení týchto úloh neodovzdajte iba hľadaný regulárny výraz, ale vysvetlite, ako ste našli daný regulárny výraz a prečo vášmu regulárnemu výrazu zodpovedajú iba správne reťazce a žiadne iné.

Pomôcka

Existuje množstvo online nástrojov, kde si viete vyskúšať prácu s regulárnymi výrazmi a testovať riešenia tejto úlohy. Medzi ne patrí napríklad aj https://regex101.com/. Ak na tejto stránke napíšete do okienka "regular expression" svoju vzorku a do okienka "tesst string" nejaký text, stránka vo vami zadanom texte *vyhľadá podreťazec*, ktorý zadanej vzorke zodpovedá. Vyskúšajte si napr. zadať vzorku „jabl(k|ck)o" a text „mak jablko jahoda". Ak si chcete vynútiť test, či celý zadaný text zodpovedá vašej vzorke, dopíšte na jej začiatok znak „^" a na jej koniec znak „$". Tieto dva špeciálne znaky predstavujú začiatok a koniec celého textu, v ktorom hľadáme.

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. V tomto kole to bude práca s tabuľkovými editormi.

3. Príhody krokodíla Karola s tabuľkami

16 bodov za riešenie

Ak máte akékoľvek otázky ohľadom tejto, úlohy napíšte Janovi Hozzovi na janoh@ksp.sk

V zadaní chýba ešte niekoľko odkazov, ktoré budú doplnené v najbližších dňoch.

Krokodíl Karol sa v poslednej dobe začal hrať s rôznymi tabuľkovými kalkulátormi. Tieto programy sú v správnych rukách veľmi mocné nástroje, ktoré umožňujú jednoduchým spôsobom spracovať väčšie množstvo údajov či dokonca niečo spočítať. Táto vlastnosť sa Karolovi veľmi pozdávala a zaujímalo by ho, čo všetko s nimi vie robiť. Zadal vám preto niekoľko úloh a je zvedavý, či sa vám ich podarí vyriešiť. Dokonca vám sľúbil aj jednobodovú prémiu, ak zvládnete úplne všetko.

Ako riešiť

Túto úlohu môžete riešiť v troch rôznych tabuľkových kalkulátoroch. Google Spreadsheet, Microsoft Excel alebo LibreOffice Calc. Asi najpreferovanejšie možnosť je používať Google Spreadsheety, lebo len tam vieme zaručiť, že dokument, ktorý dostanete, bude vyzerať tak ako má. Dokonca nepotrebujete ani Google konto, stačí, ked pôjdete na tento link prask.ksp.sk/specialne/prask/2/2/3/ a my vám vygenerujeme váš vlastný dokument na riešenie. Ak si v tomto dokumente náhodou niečo pokazíte a chcete ho vrátiť do pôvodného stavu, chodte `File -> See revision history`. Tam si zvolíte poslednú (alebo teda tú čo chcete navrátiť) revíziu a stlačíte `Restore this revision`. V prípade, že si s tým nebudete vedieť pomôcť, tak nám napíšte.

V prípade, že chcete použiť Microsoft Excel, stiahnite si tento súbor `https://people.ksp.sk/~prask/specialne/subory/excel.xlsx`, do ktorého môžete doplňať svoje riešenie. Pre LibreOffice si môžete stiahnuť tento súbor `https://people.ksp.sk/~prask/specialne/subory/libreoffice.ods`. Takisto vám odporúčame si nainštalovať najnovšiu verziu LibreOfficu (napríklad z tejto stránky: www.libreoffice.org/download/libreoffice-fresh/). V oboch prípadoch je však možné, že dokument nebude vyzerať úplne presne tak ako by sme chceli. Je možné, že sa zmenia šírky stĺpcov alebo formát textu. Ani jedna z týchto vecí by nemala byť dôležitá pri riešení a snáď ju dokážete opraviť sami. Keby sa vyskytol nejaký iný problém, kľudne sa ozvite.

Tutoriál

Ak ste nikdy nerobil s tabuľkovým kalkulátorom, je načase, aby ste sa naučil nejaké základy, ktoré vám bude treba pri riešení nasledujúcich úloh. A takisto sa vám zídu v škole, keď budete pracovať s podobným programom.

K dispozícii máte tento krátky textový tutoriál prask.ksp.sk/tutorial_tabulky, alebo sa môžete pozrieť na to, ako sa tie veci používajú priamo vo vami zvolenom programe: google spreadsheet, Excel alebo Libreoffice.

Takisto si viete nájsť dokumentáciu alebo rýchle tutoriály k ľubovoľnému z tých programov.

Odovzdávanie

Cez webové rozhranie odovzdajte súbor vo formáte `.ods` alebo `.xls` (poprípade `.xlsx`). Aj v prípade, že používate Google Spreadsheet, stiahnite svoj dokument ako `.ods` súbor[5] a normálne ho submitnite (ale budeme sa pozerať aj na webovú verziu).

Úloha

a) (1 bod) Krokodíl Karol už vie sčítavať, no násobenie mu robí ťažkosti. Vyrobte tabuľku veľkej násobilky a pomôžte tak Karolovi naučiť sa násobenie. Takto by mala vyzerať časť tabuľky. Vašou úlohou je vyplniť celú tabuľku 20×20.

	A	B	C	D	E	F	G	H	I
1		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
2	**1**	1	2	3	4	5	6	7	8
3	**2**	2	4	6	8	10	12	14	16
4	**3**	3	6	9	12	15	18	21	24
5	**4**	4	8	12	16	20	24	28	32
6	**5**	5	10	15	20	25	30	35	40
7	**6**	6	12	18	24	30	36	42	48

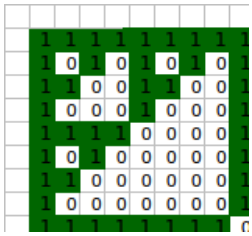
b) (2 body) Karol aj so svojimi priateľmi krokodílmi síce žije v rezervácií, ale aj tak sú vystavení mnohým nebezpečenstvám a z tohto dôvodu sa naučili efektívne brániť. Najstarší krokodíl navrhol, aby sa pri obrane postavili krokodíly do takzvanej Sierpinskeho formácie. V severozápadnom rohu (vľavo hore na

[5] File -> Download as -> OpenDocument format (`.ods`)

mape), odkiaľ prichádzajú nepriatelia, stojí najmocnejší krokodíl. Pre všetky ďalšie políčka platí, že na políčku je krokodíl, ak vedľa neho smerom na sever alebo na západ je presne jeden krokodíl.

Vytvorte tabuľku, ktorá bude odrážať reálne rozostavenie vojakov na bojisku. Nula značí, že na políčku nie je krokodíl, jednotka znamená, že je. Vyfarbite políčka s krokodílmi tmavozelenou farbou. (Rada: Skúste použiť podmienené formátovanie, po anglicky Conditional Formatting.)

Ľavá horná časť tabuľky by mala vyzerať tak ako na obrázku nižšie. Vy však vyrobte celú tabuľku 32×32.



c) (3 body) Pre každý deň posledného roku vieme, koľko peňazí minul Karol v školskom bufete a tiež vieme, či dostal nejaké vreckové. Vypočítajte koľko vreckového dostal za celý rok, koľko minul v bufete počas celého roku a koľko za celý rok ušetril. Potom spočítajte, koľko dní bol Karol zadĺžený (t.j. od začiatku roka minul viac ako dostal), koľko priemerne minul v pondelky a aký bol jeho priemerný stav účtu (teda výdavky aj príjmy dokopy). Napokon zistite pre každý mesiac, koľko peňazí ušetril v danom mesiaci. Spravte tabuľku tak, že keby Karol zmenil údaje, tak sa štatistiky automaticky prepočítajú.

d) (3 body) Máte danú výsledkovku s **Pr**ogramátorskej a **A**lgoritmickej **S**úťaže **K**rokodílov. Je to 7 stĺpcov, v ktorých je postupne. Meno, ročník, škola a počty bodov za príklady 1 až 5.

Pre každého súťažiaceho zistite celkový počet bodov za úlohy, pričom najlepšie vyriešená a najhoršie vyriešená úloha sa ráta za polovicu. Následne zvýraznite v tabuľke plné počty (Plný počet za každú úlohu je 15 bodov). Utrieďte účastníkov podľa počtu bodov a pre každého zistite jeho poradie vo výsledkovke.

Na obrázku môžete vidieť výsledky pre prvých 5 ľudí, všimnite si, že keď majú viaceri ľudia rovnaký počet bodov, majú aj rovnaké poradie vo výsledkovej listine.

Poradie	Meno a priezvisko	Roč	Prvá úl	Druhá ú	Tretia ú	Stvrtá ú	Piata úl	Spolu Bodov
1	Filip Kerák	8	15	15	12	10	13	52,5
1	Tomáš Buček	5	14	15	14	10	12	52,5
3	Jaroslav Paška	6	13	13	9	15	14	52
3	Jaroslav Varga	6	15	12	11	14	13	52
3	Tomáš Tkáčik	7	15	10	15	15	9	52

e) (3 body) Karol sa zaujíma o dávnu históriu svojho živočíšneho druhu. Zistil, že pred desaťtisíc rokmi žilo v istom údolí 100 šablozubých krokodílov, 10 mamutov a 100 000 holubov. Krokodíly sa živili holubmi a občas napadli aj nejakého mamuta. Keď bolo holubov málo, krokodíly boli hladné a ich počet ubúdal. Keď je holubov dostatok, počet krokodílov rástol. Holuby zasa znečisťujú životné prostrede, čo spôsobuje pokles počtu mamutov. A napokon sú tu klimatické zmeny, ktoré tiež spôsobujú ubúdanie mamutov.

Každý rok sa počty mamutov, holubov a krokodílov trocha zmenili. Označme si k, m a h počet krokodílov, mamutov a holubov na začiatku roka. Nech i je počet rokov, ktoré uplynuli od roku 10 000 p.n.l.

Počas roka sa vždy narodia dva mamuty a $h/200\,000$ mamutov zomrie kvôli znečisteniu, $k/2\,000$ mamutov zomrie kvôli útokom krokodílov a $i/500$ mamutov zomrie kvôli globálnemu otepľovaniu.

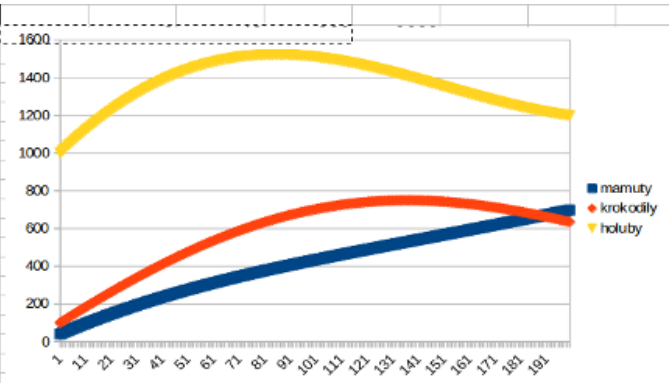
Narodí sa $5\sqrt{h}$ holubov a $3k$ holubov je zožratých krokodílmi.

Napokon 1 percento krokodílov zomrie na starobu, $m/20$ krokodílov je zašliapnutých mamutom a narodí sa $h/10\,000$ nových krokodílov.

Zistite, v ktorom roku vyhynú mamuty. Spočítajte množstvo krokodílov, mamutov a holubov až do toho roku. Vykreslite tieto počty v prehľadnom grafe. (Počet mamutov v grafe prenásobte štyrmi a počet holubov predeľte 100). Vo výpočtoch sa netrápte tým, že počet živočíchov nemusí byť celé číslo. Áno, na svete môže byť aj 123.456 krokodílov.

Na obrázku môžeme vidieť správne počty krokodílov v prvých 18 rokoch a tiež graf, znázorujúci počty počas prvých 200 rokov.

Počet rokov	Počet mamutov	Počet krokodílov	Počet holubov
1	10	100	100000
2	11,448	108,5	101281,13883
3	12,8833443058	116,970713883	102546,873719
4	14,3061245803	125,4115269008	103797,108617
5	15,7164332738	133,8218162644	105031,751969
6	17,1143636058	142,200951635	106250,716648
7	18,5000095467	150,5482956032	107453,919891
8	19,8734657995	158,8632041589	108641,28324
9	21,2348277812	167,1450271514	109812,732487
10	22,5841916052	175,3931087395	110968,197617
11	23,9216540627	183,6067878336	112107,612762
12	25,247312605	191,7853985283	113230,916145
13	26,561265325	199,9282705272	114338,050039
14	27,8636109396	208,0347295596	115428,960721
15	29,1544487712	216,1040977891	116503,598425
16	30,4338787302	224,1356942152	117561,917305
17	31,7020012965	232,128835067	118603,875391
18	32,958917502	240,0828341906	119629,434554
19	34,2047289122	247,997003429	120638,560466

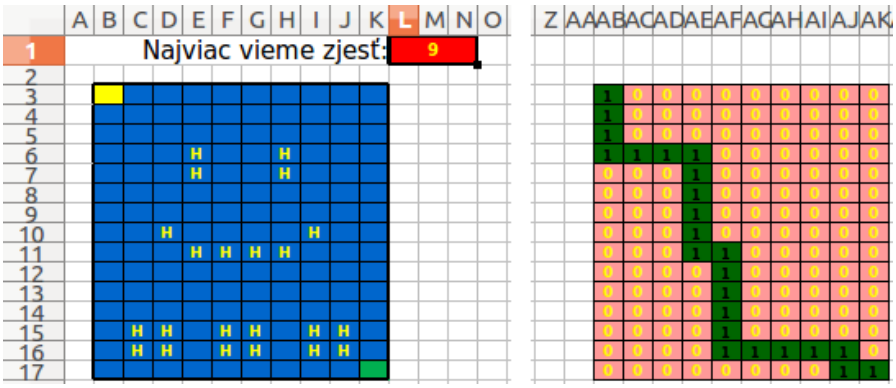


f) (3 body + 1 bonusový bod) Pri rozmýšľaní o holuboch krokodíl Karol poriadne vyhladol. A tak sa vybral na námestie pochytať nejaké holuby. Námestie má rozmery 32×22 dlaždíc a na niektorých dlaždiciach sedia holuby. Karol stojí v severozápadnom rohu námestia a môže sa hýbať len na juh a na východ. Keď príde na dlaždicu, na ktorej sedí holub, okamžite ho zožerie. Koľko najviac holubov dokáže zjesť?

Úlohu riešte tak, že keď sa zmení umiestnenie holubov na kachličkách, tak sa automaticky prepočíta odpoveď.

Bonus: Graficky (pomocou jednotiek a núl, prípadne pomocou farby) vykreslite trasu, na ktorej zje Karol najväčší možný počet holubov. Ak je takých ciest viac, vykreslite ľubovoľnú.

Na obrázku nižšie môžete vidieť, ako by vyzeralo riešenie pre menšie námestie s iným rozmiestnením holubov.



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 cykly a druhá na polia 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**.

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. Perníková optimalizácia

0 bodov za riešenie

Táto úloha sa dá nahradiť riešením sady ***loops_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 Matúšovi Zeleňákovi na matus@ksp.sk

Znova raz nastal čas pečenia medovníkov a tak Žaba vytiahol z kredenca svoje tri formičky a začal vykrajovať! Po chvíli si však uvedomil, že zhŕňať odrezky z cesta a stláčať ich dokopy bude strašne otravné a neefektívne a preto by rád vedel, koľko perníkového cesta má urobiť tak, aby sa z neho dalo vykrajovať bez nechávania zvyškov.

Žaba je naviac strašne lenivý a v žiadnom prípade nechce následne umývať viac ako jednu zo svojich formičiek. Zároveň vie, že sa mu do robenia medovníkov neoplatí pustiť, ak plánuje pripraviť menej ako, povedzme, 500 gramov medovníkového cesta. A je rád, keď má na výber, preto by chcel vedieť niekoľko prvých možností pre množstvo cesta, ktoré bude vyhovovať jeho požiadavkam. Dúfam, že perfekcionistu Žabu nenecháte vyhladovať a pomôžete mu s jeho problémom...

Úloha

Viete, že Žaba musí spraviť aspoň n gramov cesta, aby sa mu do pečenia vôbec chcelo pustiť, a zároveň musí byť dané množstvo cesta deliteľné veľkosťou **aspoň jednej** z formičiek veľkostí a, b a c (inak by vznikali nechcené zvyšky). Vašim cieľom je nájsť prvých m najmenších množstiev cesta, ktoré sú väčšie alebo rovné ako n a túto podmienku spĺňajú.

Váš program bude testovaný na piatich sadách vstupov. Pre každú sadu nájdete obmedzenia vstupných hodnôt v tabuľke nižšie. Body dostanete za každú sadu, ktorú úspešne vyriešite. Pozor, posledné dve sady obsahujú naozaj veľké čísla, riešenia používajúce klasické 32-bitové premenné typu `int` alebo `longint` na nich zaručene zlyhajú. Použite preto 64-bitové premenné – typ `long long` v C++, typ `int64` v Pascale alebo typ `long` v Jave.

Číslo sady	1	2	3	4	5
$n \leq$	10 000	10^6	10^9	10^{12}	10^{15}
$m \leq$	100	1 000	20 000	10^5	10^6
$a, b, c \leq$	100	1 000	20 000	10^5	10^6

Malá rada na záver: existuje vzorové riešenie, ktoré nepoužíva žiadne polia. Ak aj úlohu vyriešite s ich použitím, skúste napísať aj program, ktorý ich nepoužije.

Vstup

Na prvom riadku vstupu dostanete dve celé čísla n a m, pričom n označuje minimálne množstvo cesta a m je počet možností, ktoré máte nájsť. Druhý riadok vstupu obsahuje tri čísla a, b, c určujúce veľkosti troch formičiek, ktoré má Žaba k dispozícii.

Výstup

Vypíšte jeden riadok s m číslami oddelenými medzerou označujúcimi vyhovujúce množstvá cesta usporiadané od najmenšieho po najväčšie. Nezabudnite na znak konca riadku.

Príklad

vstup

```
7 9
3 5 11
```

výstup

```
9 10 11 12 15 18 20 21 22
```

Čísla 9, 12, 15, 18 a 21 sú deliteľné 3, čísla 10, 15 a 20 číslom 5 a čísla 11 a 22 číslom 11. Vynechané čísla 13, 14, 16, 17, 19 nie sú deliteľné ani jedným z týchto čísel, preto sa vo výsledku nevyskytli.

vstup

```
42 6
1 2 3
```

výstup

```
42 43 44 45 46 47
```

Formičku 1 vieme použiť na akokoľvek veľké cesto.

5. Poradie tanečníkov

0 bodov za riešenie

Táto úloha sa dá nahradiť riešením sady ***arrays_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 Romanovi Sobkuliakovi na r.sobkuliak@gmail.com

Jednou z najdôležitejších častí stužkovej je úvodný valčík. Okrem samotného tanca by tiež všetci mali vedieť, kde je ich miesto pri nástupe. To si zapamätajú jednoducho – stačí vedieť, ktorý spolužiak stojí pred nimi.

Prvou v rade je takmer vždy Laura, ktorá má celú stužkovú pod palcom. Laura práve hysterčí. Je deň pred veľkým večerom a ona nevie poradie nástupu! Napísala už všetkým spolužiakom a od každého z nich zistila, kto stojí pred ním. Potrebuje už len z nazbieraných informácii zistiť pôvodné poradie. Musí ale do zajtra ešte vybaviť veľa vecí a nemá čas. Nepomohli by ste jej?

Úloha

V triede je n žiakov očíslovaných 1 až n. O $n-1$ žiakoch dostanete informáciu o tom, kto sa nachádza pred nimi. Vašou úlohou je zistiť poradie jednotlivých žiakov.

Vstup

Na prvom riadku vstupu je číslo n udávajúce počet študentov. Každý z nasledujúcich $n-1$ riadkov obsahuje popis jedného žiaka. Každý riadok obsahuje dve medzerou oddelené čísla x a y, ktoré znamenajú, že pri nástupe sa pred žiakom s číslom x nachádza žiak s číslom y.

Výstup

Výstup bude obsahovať n riadkov. Na i-ty z nich napíšete číslo i-teho žiaka v poradí pri nástupe.

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. Naviac, pre sady 1, 2 a 4 platí, že na prvom mieste výslednej rady je číslo 1.

Číslo sady	1	2	3	4	5
maximálne n	50	1 000	1 000	1 000 000	1 000 000

Príklady

vstup

```
1
```

výstup

```
1
```

Jediný žiak musí byť vždy prvý. Všimnite si tiež, že sme načítali iba jedno číslo zo vstupu.

vstup

```
5
5 2
4 3
2 4
3 1
```

výstup

```
1
3
4
2
5
```

vstup

```
6
2 3
4 6
5 1
1 2
6 5
```

výstup

```
3
2
1
5
6
4
```

Keďže na prvom mieste rady je číslo 3*, takýto vstup sa nemôže objaviť v sadách 1, 2 a 4.*