PRASK

Leták zimnej časti IV. 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.

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.

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

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

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
IV. ročník, 2017/18
Katedra základov a vyučovania informatiky FMFI UK,
Mlynská Dolina, 842 48 Bratislava

Úlohy 1. kola zimnej časti

Termín odoslania riešení tejto série je pondelok **30. október 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. Premiestnení faraóni

15 bodov za riešenie

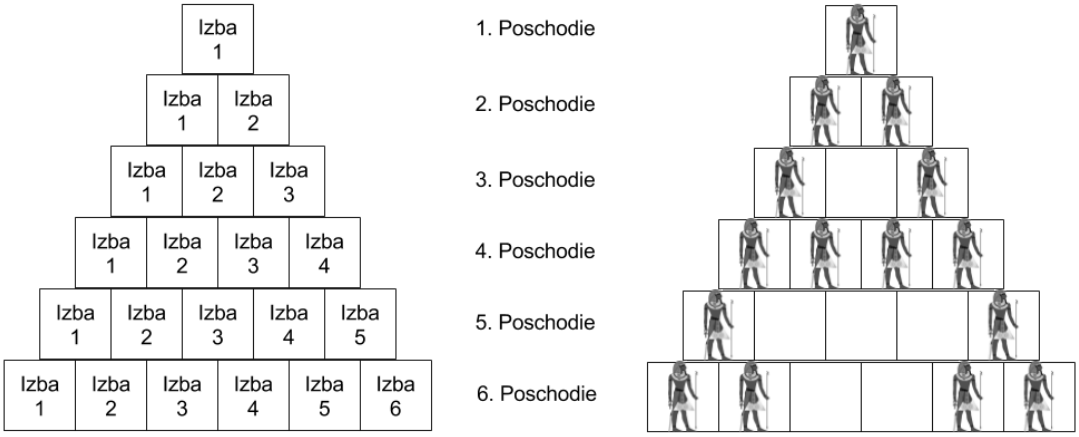
Ak máte akékoľvek otázky ohľadom tejto úlohy, napíšte Andrejovi na andrejkorman1@gmail.com

Egypťania sa tento rok rozhodli spraviť rozsiahlu rekonštrukciu pyramíd. Potrebujú preto premiestniť všetkých faraónov do jednej veľkej spoločnej pyramídy. Na samý vrch tejto pyramídy uložia ich najobľúbenejšieho faraóna a zvyšných dajú do miestností pod ním. Keďže nová pyramída ešte nestojí a my nevieme, koľko poschodí bude mať, poschodia budeme číslovať od vrchu, začínajúc poschodím číslo 1.

Ich cieľom je uložiť do tejto pyramídy čo najviac faraónov. Rozhodli sa preto, že len čo ju postavia, začnú ju zvrchu postupne napĺňať. Tu však narazili na dva problémy. Z dôvodu hmotnosti starovekých sarkofágov môžu do ktorejkoľvek miestnosti vložiť faraóna, len ak sa priamo nad ním nenachádzajú dvaja iní faraóni. Navyše faraóni sú spoločenské bytosti, preto musí priamo nad každým faraónom (s výnimkou hlavného faraóna na vrchu pyramídy) ležať aspoň jeden iný faraón.

Postup napĺňania teda vyzerá nasledovne: prechádzame postupne po poschodiach všetky izby, pričom začíname najvyšším poschodím. Ak sa priamo nad danou izbou nachádzajú dvaja faraóni, alebo tam naopak žiaden faraón nie je, izba ostane prázdna. V opačnom prípade (teda priamo nad izbou je jeden faraón) tam vložíme ďalšieho faraóna.

Na nasledujúcom obrázku je príklad pyramídy so šiestimi poschodiami, ktorá bola zaplnená práve takýmto spôsobom.



Vidíme, že v izbe 2 na 3. poschodí nie je umiestnený faraón a to preto, že obe izby nad ním sú plné. Toto nastalo aj v troch izbách na 5. poschodí. V izbe 3 na 6. poschodí taktiež nie je faraón a to preto, že mu nad ním v izbách 2 a 3 na 5. poschodí chýba spoločnosť. Toto platí aj o izbe 4 na 6. poschodí.

V izbe 2 na štvrtom poschodí však faraón je, lebo nad ním je práve jeden iný faraón (izba 1, 3. poschodie). Takisto si všimnite, že v izbe 1 je na každom poschodí faraón, lebo nad ním môže byť iba jeden faraón a ten tam vždy je.

Uvedeným spôsobom vieme vyplniť ľubovoľne veľkú pyramídu. Odporúčame vám si to vyskúšať napríklad na papier a nakresliť si, ako by vyzerala pyramída, ktorá by mala o niečo viac poschodí.

Úloha

a) (4 body) Na obrázku vyššie môžeme vidieť, že na 4. poschodí ja každá izba obsadená faraónom. Ešte pred postavením pyramídy by architektov zaujímalo, či je počet takýchto plne obsadených poschodí obmedzený alebo nie. Vyššie uvedenú pyramídu by sa totiž rozhodne neoplatilo postaviť 3 poschodovú keďže práve 4. poschodie vieme celé naplniť.

   Vašou úlohou je zistiť, či sa podobný riadok v pyramíde vyskytuje aj na spodnejších poschodiach. Pre zjednodušenie si predstavte, že túto pyramídu sme postavili nekonečne vysokú. Platí, že sa tam pri plnení zhora vždy po istej dobe takéto poschodie plné faraónov znova vyskytne? Ak áno, vedeli by ste nájsť pravidlo, ktoré určí, či poschodie s poradovým číslom x je práve takéto poschodie? Obe svoje tvrdenia sa pokúste dokázať.

b) (6 bodov) Akonáhle bude pyramída dostavaná, bude staviteľov zaujímať, či do konkrétnej izby majú niekoho nasťahovať alebo nie. Z tohto dôvodu si musia byť istí, že viete o každej izbe správne určiť, či bude alebo nebude obývaná. Inak vám takúto dôležitú úlohu do rúk určite nemôžu zveriť.

   Vašou úlohou je zistiť, či bude izba číslo 10 na 20. poschodí obývaná alebo nie. Toto isté zistite aj o izbe číslo 266 na poschodí 276. K vašej odpovedi by ste mali napísať aj popis vášho riešenia, teda ako ste sa k tomuto výsledku dopracovali.

c) (5 bodov) Po preverení vašich schopností sa už môžete pustiť do práce. Pyramída sa medzičasom postavila a vy ste dostali za úlohu vymyslieť postup, vďaka ktorému vieme o ľubovolnej izbe povedať, či bude alebo nebude obývaná. Egypťanom však došiel papyrus a tak by boli veľmi radi, ak by ste vymysleli šetrnejší spôsob ako kreslenie si celej pyramídy.

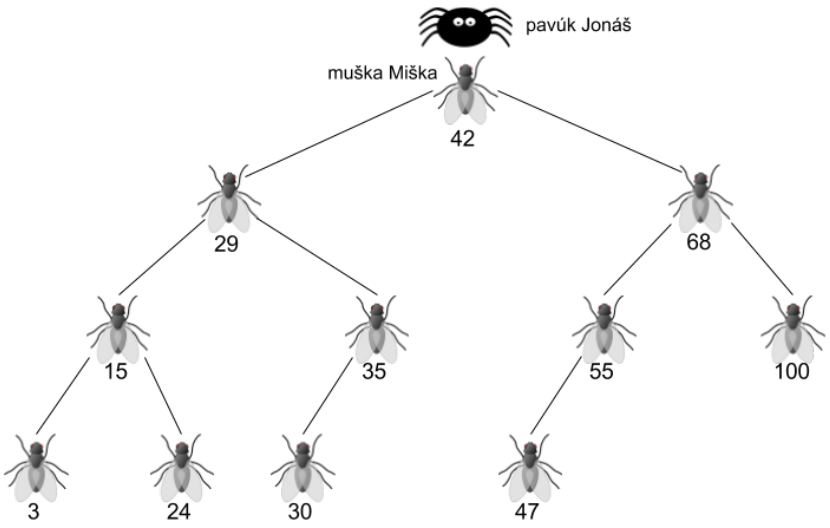
   Vašou úlohou je vymyslieť postup, ktorý určí pre ľubovolné poschodie a izbu na tomto poschodí, či v nej bude alebo nebude faraón a to v čo najmenšom počte krokov. Vzorové riešenie by bolo schopné zistiť zaplnenosť ktorejkoľvek izby aj na poschodí s číslom rádovo okolo 1 000 000 (a aj omnoho viac) a to len s použitím papiera a rozumného množstva času. Upozorňujeme, že okrem vysvetlenia tohto postupu na nejakom príklade musíte uviesť jednoznačný postup, ktorý vie opravovateľ na ktoromkoľvek inom poschodí a izbe zopakovať a dopracovať sa k správnemu výsledku.

2. Pavúčí sklad

15 bodov za riešenie

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

Pavúk Jonáš nie je ako zvyšné pavúky. Rád programuje, je poriadkumilovný a tká vskutku nezvyčajné siete, do ktorých si potom ukladá potravu. To mu však občas dosť komplikuje život. Navigovať sa v jeho sieťach nie je vôbec jednoduché a odkedy si robil zásoby na zimu, stratil prehľad o všetkých chytených muškách. Našťastie pri stavaní siete dodržiaval niekoľko prísnych pravidiel, a preto sa v nej určite bude dať nájsť nejaký systém. Pomôžete mu s tým?



Na obrázku môžete vidieť jeden možný príklad siete pavúka Jonáša. Teraz vám povieme, aké pravidlá Jonáš dodržiava pri stavaní siete. Môžete si teda overiť, že aj táto sieť spĺňa dané pravidlá.

V sieti sa nachádza niekoľko mušiek, ktoré Jonáš chytil a zviazal do pavučiny. Každá z týchto mušiek má rôznu váhu reprezentovanú jedným celým číslom. Zvyšok pavučiny tvoria vlákna naťahané medzi týmito muškami. Na samom vrchu pri Jonášovi je prvá muška, ktorú chytil – muška Miška. Z mušky Mišky vedú dodola najviac dve vlákna, jedno vedie doľava a druhé doprava. Naviac musí platiť, že všetky mušky, ktoré sú na ľavej strane **sú ľahšie ako Miška** a všetky mušky, ktoré sú napravo **sú ťažšie ako Miška**.

Táto vlastnosť však neplatí iba pre mušku Mišku, ale pre všetky mušky v sieti. Teda nech si zoberieme ľubovoľnú mušku, tak z nej dodola vedú najviac dve vlákna (niekedy len jedno ako pri muške 35 a niekedy dokonca žiadne ako pri muške 24) a všetky mušky, ktoré sú pod ňou na ľavej strane sú ľahšie ako ona a všetky mušky, ktoré sú pod ňou na pravej strane sú ťažšie ako ona.

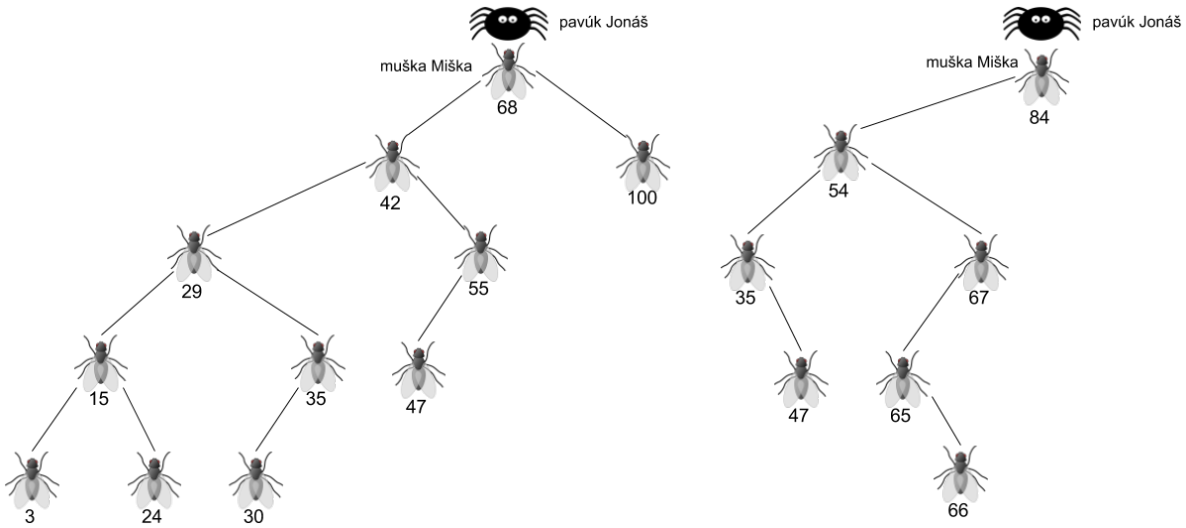
Na obrázku si môžete skontrolovať, že to naozaj tak je. Ak si zoberieme mušku 29, tak naľavo pod ňou sú mušky 3, 15 a 24, ktoré sú všetky ľahšie a napravo pod ňou sú mušky 35 a 30, ktoré sú ťažšie.

Jonáš sa po sieti môže pohybovať iba po naťahaných vláknach. Napríklad ak by sa chcel dostať k muške 30, musel by najskôr zliezť k muške 29, potom k muške 35 a až potom by sa vedel dostať k muške 30. Pri tom by zliezol po troch vláknach.

Úloha

Táto úloha sa skladá z niekoľkých podúloh. V každej z nich je vašou úlohou prísť s postupom, ktorý pomôže Jonášovi dosiahnuť jeho cieľ. Tento postup by mal byť všeobecný a mal by fungovať na ľubovolnej sieti, ktorá spĺňa podmienky zadania. K dispozícii máte tri obrázky rôznych sietí. Overte si, či váš postup funguje na každej z nich. A možno si nakreslite aj ďalší obrázok, ktorý bude vyzerať ešte inak a bude obsahovať iné čísla a overte si to aj na ňom.

Ku každej podúlohe, ktorú vyriešite, napíšte postup, ktorým sa má Jonáš riadiť. Takisto popíšte, prečo tento postup funguje pri ľubovoľnej sieti a môžete tiež ukázať, ako sa tento postup dá použiť na jednom z obrázkov zo zadania.



a) (2 body) Jonáš stojí na vrchu siete pri muške Miške. Chcel by zjesť najmenšiu mušku, ktorú má v sieti chytenú. Navrhnite postup, ktorým sa dostane k najľahšej muške. Snažte sa, aby pri tom musel zliezť po čo najmenej vláknach.

   Pri sieti z prvého obrázka by teda Jonáš mal nájsť mušku s váhou 3. Nezabudnite zdôvodniť, prečo váš postup naozaj nájde vždy tú najľahšiu mušku.

b) (2 body) Jonáš stojí pri muške Miške. Chcel by nájsť najľahšiu mušku, ktorá je ťažšia ako muška Miška. Viete mu pomôcť navrhnúť postup, ktorým takúto mušku nájde?

   V prvej sieti, v ktorej má muška Miška váhu 42, by teda mal nájsť mušku s váhou 47.

c) (3 body) V tejto podúlohe by chcel Jonáš zjesť mušku s veľmi konkrétnou váhou. Jonáš vám túto váhu zadá, označme si ju x. Vymyslite postup, ktorým Jonáš nájde mušku s váhou x (pre ľubovoľné x, ktoré vám Jonáš povie), alebo zistí, že taká muška v sieti nie je. Opäť sa snažte, aby sa Jonáš čo najmenej nachodil. Aj v tejto úlohe Jonáš začína na vrchu siete.

Ak by teda v prvej sieti chcel zjesť mušku s váhou 30, váš postup by mal nájsť cestu po troch vláknach cez mušky 29 a 35. Ak by ale hľadal mušku s váhou 50, váš postup by mal zistiť, že taká muška tam nie je. Aj pri tom môže Jonáš chodiť po sieti.

d) (4 body) Jonášovi sa pri dnešnom love darilo a chytil mušku s váhou x. Víťazoslávne ju teraz drží na vrchu siete. Chcel by si ju však niekde zavesiť. To však nemôže spraviť len tak. Chce aby výsledná sieť naďalej spĺňala všetky podmienky zo zadania. A naviac, nechce pretrhávať žiadne už existujúce vlákna. Novú mušku preto môže zavesiť iba pod mušku, z ktorej nevedie naľavo alebo napravo vlákno. Nájdite riešenie, pri ktorom sa Jonáš čo najmenej nachodí.

Napríklad ak by ulovil mušku s váhou 58, mohol by ju zavesiť napravo od mušky 55. Nemohol by ju však zavesiť pod mušku 3 (či už vľavo alebo vpravo), lebo takéto zavesenie by porušilo jednu zo základných podmienok – naľavo od mušky 15 by bola ťažšia muška 58. Takisto túto mušku nemôže zavesiť pod mušku 29, keďže tá už má mušku naľavo aj napravo od seba. Všimnite si, že toto platí rovnako pre prvý aj druhý obrázok.

Ak by Jonáš ulovil mušku s váhou 1, správne zavesenie na týchto obrázkoch by bolo naľavo od mušky 3.

e) (4 body) Jonáš zjedol jednu z mušiek a teraz stojí na jej mieste. Ak by však na toto miesto nedal žiadnu inú mušku, sieť sa mu pravdepodobne pokazí. Chcel by preto previazať niektoré vlákna tak, aby opäť dostal sieť, ktorá vyhovuje jeho podmienkam. Pomôžte mu nájsť postup, ktorým také niečo vie spraviť. Snažte sa, aby pri vašom postupe musel preväzovať čo najmenej vlákien.

Pozrieme sa na prvú sieť z druhého obrázku. Ak Jonáš zje mušku s číslom 24, vlastne nemusí nič robiť, sieť spĺňa zadané pravidlá. Pri zjedení mušky 55 by to bolo o niečo ťažšie, stále by to však vedel pomerne rýchlo previazať. Ak by však zjedol napríklad mušku 29, situácia by bola výrazne ťažšia.

Praktická úloha

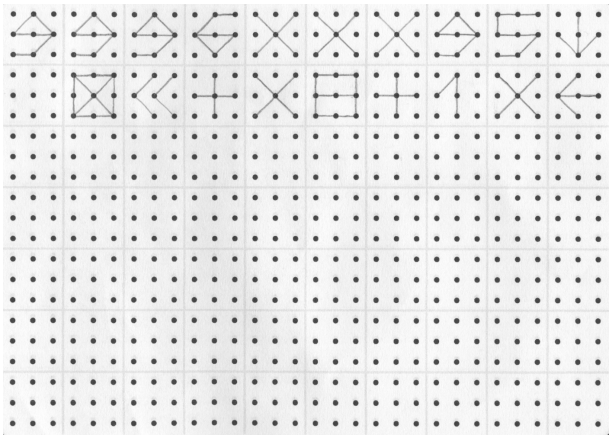
Pri práci s počítačom je potrebné vedieť pracovať aj s rôznymi nástrojmi, ktoré slúžia na úpravu obrázkov, prácu so zvukom či vyhľadávaním na internete. V tejto časti ťa preto zakaždým čaká nejaká netradičná úloha.

3. Pohodlné Kreslo

15 bodov za riešenie

Ak máte akékoľvek otázky ohľadom tejto úlohy, napíšte Dávidovi na davidb@ksp.sk

Tomi išiel domov zo školy, a rozmýšľal, čo by robil, ak by sa dostal na opustený ostrov. Založil by si oheň, ulovil ryby, postavil príbytok, ... Keď si už myslel, že má všetko premyslené a nič ho neprekvapí, uvedomil si, že by nemal ako programovať. A keby si aj začal písať programy na drevo, nemal by ako zistiť, či niekde neurobil chybu. Tomi by bez programovania neprežil, a preto si povedal, že vymyslí nový programovací jazyk, ktorý bude pohodlný, bude sa ľahko kresliť na drevo a bude sa dať ľahko ručne simulovať jeho vykonávanie. Nazval ho Kreslo (pretože sa kreslí a je pohodlný). Tomi by rád vedel, ako dobrý je jeho jazyk a čo všetko sa v ňom dá naprogramovať. Na to potrebuje vašu pomoc.



Úloha

Kreslo je veľmi jednoduchý jazyk. Pracuje iba s celými číslami, ktoré vie načítavať zo vstupu a vypisovať na výstup. Aby sa Tomimu ľahko kreslilo, celý program je nakreslený do mriežky a každá inštrukcia zaberá jeden štvorec. Podrobnosti o tom, ako tento jazyk funguje a ako v ňom písať programy nájdete na ksp.sk/~prask/specialne/4/1/3/manual.pdf

My našťastie nie sme na opustenom ostrove, a preto môžeme používať elektronický editor, ktorý dokáže aj simulovať beh programu. Nájdete ho spolu s ukážkovými programami na adrese ksp.sk/~prask/specialne/4/1/3/.

Vašou úlohou bude vyriešiť nasledujúce podúlohy a odovzdať ZIP súbor, ktorý obsahuje zdrojové kódy vašich riešení (dajú sa stiahnuť z editora). ZIP by mal obsahovať súbory `1.kr` až `5.kr` s vašimi riešeniami jednotlivých podúloh.

Podúlohy

a) (1 bod) Na vstupe je zadané číslo a ($1 \leq a \leq 1\,000$). Vypíšte číslo 7 ak a je deliteľné číslom 7 (je to násobok sedmičky), inak vypíšte číslo a nezmenené.

vstup	výstup
`21`	`7`
`47`	`47`

b) (3 body) Na vstupe je zadané číslo n ($1 \leq n \leq 100$). Vypíšte čísla od n do 1 v klesajúcom poradí.

vstup	výstup
`7`	`7 6 5 4 3 2 1`

c) (4 body) Na vstupe sú zadané čísla x a y ($0 \leq x \leq y \leq 100$). Vypíšte v rastúcom poradí všetky čísla od x do y (vrátane), ktoré sú násobkami čísla 3.

vstup	výstup
`39 47`	`39 42 45`

d) (4 body) Na vstupe je zadané číslo n ($1 \leq n \leq 1000$). Vypíšte najmenšie číslo a, $1 < a \leq n$ ktoré delí n (zvyšok po delení n číslom a je 0).

vstup	výstup
`35`	`5`

Deliteľmi čísla 35 sú čísla 1, 5, 7 a 35. Keďže 1 vypísať nemôžeme, najmenší deliteľ je číslo 5.

e) (3 body) Prvočísla sú prirodzené čísla väčšie ako 1, ktoré nie sú deliteľné žiadnym číslom od 2 po $p - 1$. Napríklad čísla 2, 5, 7, 23 sú prvočísla, čísla 1, 4, 50, 99 nie sú.

Na vstupe je zadané číslo n ($2 \leq n \leq 100$). Vypíšte všetky prvočísla z rozsahu od 2 po n v rastúcom poradí.

vstup	výstup
`11`	`2 3 5 7 11`

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://liahen.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 budeš môcť za body riešiť ďalšie dve sady z Liahne.

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

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

4. Pixelová výzva

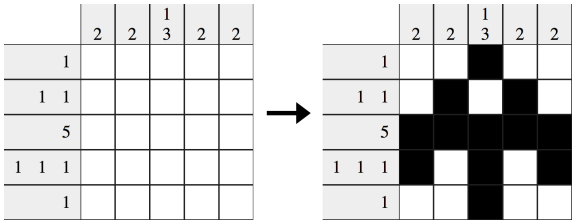
15 bodov za riešenie

Ak nevieš programovať, nezúfaj! Môžeš sa to naučiť a ešte za to získať body, ktoré sa ti budú počítať namiesto tejto úlohy.

Stačí, že pôjdeš na stránku Programátorskej Liahni (liahen.ksp.sk). Keď budeš riešiť sadu **variables_cpp**, bodmi, ktoré získaš si môžeš nahradiť riešenie tejto úlohy. Stačí ak na spodku tejto stránky odovzdáš pdf-ko s prezývkou, ktorú používaš na Liahni.

Ak máte akékoľvek otázky ohľadom tejto úlohy, napíšte Romanovi na r.sobkuliak@gmail.com

Prišli prázdniny a Bonifác sa nudil. Všetko sa ale zmenilo, keď na záchode otvoril časopis a na poslednej strane objavil maľovanú krížovku. Na nasledujúcom obrázku je jej zadanie aj s Bonifácovým riešením:

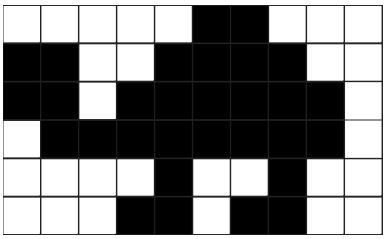


Skúste si všimnúť závislosť medzi číslami na okrajoch a výsledným obrázkom – v každom riadku je presne toľko vyfarbených úsekov (súvislých čiernych políčok), koľko je pre daný riadok za okrajom nápovedných čísel. Tieto čísla zároveň presne určujú, koľko bude daný úsek obsahovať vyfarbených políčok a tiež poradie týchto úsekov. Táto vlastnosť platí rovnako aj pre stĺpce[3].

Bonifác rýchlo prišiel na to, ako maľované krížovky riešiť, netrvalo preto dlho a vyriešil všetky, ktoré doma našiel. Teraz však potrebuje nové krížovky, pretože inak ho znovu pochytí hrozná nuda. Jeho sestra Amálka je maliarka a tak sa rozhodla, že pre Bonifáca namaľuje nové obrázky. Celý deň pilne kreslila a podarilo sa jej vytvoriť slona, stoličku, vlak či korytnačku. Z jej obrázkov je ale potrebné ešte vytvoriť zadanie maľovaných krížoviek pre Bonifáca. S touto úlohou si už nevie dať rady a tak potrebuje vašu pomoc.

Na jednom z Amálkiných obrázkov si teda ukážeme, čo bude vašou úlohou.

Úloha



[3] Vyriešiť maľovanú krížovku si môžete vyskúšať aj vy, nepotrebujete k tomu ani časopis, existuje mnoho webových stránok s týmto hlavolamom.

Ako by vyzeralo zadanie maľovanej krížovky pre tento obrázok? Potrebujeme popísať súvislé úseky v jednotlivých riadkoch a stĺpcoch. Začneme riadkami – v prvom riadku je iba jeden úsek veľkosti 2. Nasleduje riadok, ktorý by sme popísali nápovedou 2, 4, pretože prvý úsek v ňom má veľkosť 2 a druhý veľkosť 4. Analogicky postupujeme pre ostatné riadky.

Pre stĺpce postupujeme rovnako – v prvom stĺpci je iba jeden úsek veľkosti 2, v druhom stĺpci jeden úsek veľkosti 3, atď..

Na vstupe máme Amálkin čiernobiely obrázok, ktorý má r riadkov a s stĺpcov. Obrázok je mriežka s rozmermi $r \times s$, ktorá obsahuje 1 na pozíciach, ktoré sú vyfarbené, teda čierne a 0 na nevyfarbených, bielych políčkach. Vašou úlohou je z tohto obrázka vytvoriť zadanie pre maľovanú krížovku, teda spočítať súvislé úseky vyfarbených políčok v jednotlivých riadkoch a stĺpcoch.

Formát vstupu

Na prvom riadku dostanete dve čísla r a s – počet riadkov a stĺpcov obrázka. Nasleduje popis obrázka. Na každom z nasledujúcich r riadkov sa bude nachádzať s medzerou oddelených čísel. Číslo je buď 1 alebo 0, podľa toho, či je dané políčko čierne alebo biele.

Formát výstupu

Ako prvé budeme vypisovať úseky v riadkoch maľovanej krížovky. Pre každý z r riadkov obrázka, začínajúc najvrchnejším, vypíšte na samostatnom riadku medzerami oddelené veľkosti jednotlivých úsekov v danom riadku v poradí zľava doprava. Ak sa v tomto riadku žiaden úsek nenachádza, vypíšte 0.

Hneď za popisom riadkov bude nasledovať práve **jeden prázdny riadok**. Potom nasleduje popis stĺpcov. Ten bude rovnaký ako pri riadkoch – pre každý z s stĺpcov, začínajúc najľavším, vypíšte na samostatnom riadku medzerami oddelené veľkosti jednotlivých úsekov v danom stĺpci v poradí zhora nadol. Ak v stĺpci nie je žiadny vyfarbený úsek, vypíšte 0.

Hodnotenie

Pri testovaní bude vaše riešenie spustené na rôznych sadách vstupov, ktoré sa navzájom líšia obtiažnosťou ich riešenia. Za každú úspešne vyriešenú sadu dostanete 3 body. Obmedzenia pre r a s nájdete v nasledujúcej tabuľke.

Číslo sady	1	2	3	4	5
maximálne r	1	1 000	20	200	1 000
maximálne s	1 000	1	20	200	1 000

Príklady

vstup

```
6 10
0 0 0 0 0 1 1 0 0 0
1 1 0 0 1 1 1 1 0 0
1 1 0 1 1 1 1 1 1 0
0 1 1 1 1 1 1 1 1 0
0 0 0 1 0 0 0 1 0 0
0 0 1 1 0 0 1 1 0 0
```

výstup

```
2
2 4
2 6
8
1 1
2 2


2
3
1 1
4
3
4
4 1
5
2
0
```

Vstup je Amálkin obrázok, ktorý sme si ukázali vyššie. Vo výstupe je 7. riadok prázdny – oddeľuje popis

riadkov a stĺpcov. Posledný stĺpec obrázka je nevyfarbený, čomu vo výstupe zodpovedá 0 na konci.

vstup

```
1 8
1 0 1 1 1 0 1 1
```

výstup

```
1 3 2


1
0
1
1
1
0
1
1
```

Vstup, ktorý sa môže vyskytnúť v prvej sade.

5. Problematický tréning

15 bodov za riešenie

Ak nevieš programovať, nezúfaj! Môžeš sa to naučiť a ešte za to získať body, ktoré sa ti budú počítať namiesto tejto úlohy.

Stačí, že pôjdeš na stránku Programátorskej Liahni (liahen.ksp.sk). Keď budeš riešiť sadu **variables_cpp**, bodmi, ktoré získaš si môžeš nahradiť riešenie tejto úlohy. Stačí ak na spodku tejto stránky odovzdáš pdf-ko s prezývkou, ktorú používaš na Liahni.

Ak máte akékoľvek otázky ohľadom tejto úlohy, napíšte Andrejovi na andrejkorman1@gmail.com

Zima sa blíži a Syseľ by sa pomaly chcel pobrať na zimný spánok[4]. Ručička váhy sa však zastavila na vysokom čísle. Rozhodol sa teda, že bude trénovať, aby si pred tuhou zimou zlepšil kondičku. Rýchlo však zistil, že bezcieľne behanie po lese vôbec nie je efektívne. Nuž obrátil sa na Romana, profesionálneho bežca, ktorý mu ale tajomstvo svojho bežeckého umu nechcel prezradiť. Pri stretnutí s Romanom sa mu však podarilo zapamätať si časť jeho tréningového plánu.

Ten tvorí niekoľko po sebe idúcich čísel, ktoré symbolizujú zmeny v nadmorskej výške. Roman si pri behaní značí každú nezanedbateľnú zmenu v nadmorskej výške ako celé číslo. Kladným číslom označuje stúpania a záporným klesania. Pre názornú ukážku môže jeho plán vyzerať napríklad takto: `[40, 20, -25, 5]`. Vidíme, že ho na trase najskôr čakajú 2 stupáky (40 a 20), kedy dokopy stúpne o 60 výškových metrov, potom nasleduje jedno klesanie o 25 metrov a nakoniec malý kopček.

Syseľ má teraz v hlave nejakú časť Romanovho tréningového plánu, no stále nevie, aký úsek tohto plánu Roman za deň absolvuje. Uvedomil si ale jednu vec – Roman obľubuje bežať okruhy. Rád začína aj končí pri svojom dome, teda v rovnakej nadmorskej výške. Sysľovi preto stačí nájsť postupnosť, ktorej celková zmena v nadmorskej výške je nulová. V príklade vyššie by to bola časť `[20, -25, 5]`.

Takýchto postupností môže samozrejme existovať viac. Ak ale nájde najdlhšiu takúto postupnosť, bude vedieť koľko najmenej musí zabehnúť, aby sa vyrovnal Romanovi. Čísel, ktoré si Syseľ zapamätal, je však veľa a preto vás požiadal o pomoc.

Úloha

Vašou úlohou je napísať program, ktorý na vstupe dostane postupnosť n čísel, reprezentujúcu časť Romanovho plánu, ktorú si Syseľ stihol zapamätať. Na výstup potom vypíše dĺžku, začiatok a koniec **najdlhšej** podpostupnosti, ktorej celková zmena nadmorskej výšky je nulová. V danej postupnosti teda hľadáte najdlhší úsek za sebou idúcich čísel, ktoré keď sčítate dokopy, dostanete výsledok 0.

Ak existuje takýchto najdlhších podpostupností viac, vypíšte ľubovoľnú z nich.

Formát vstupu

Na prvom riadku je číslo n – dĺžka postupnosti zapamätaných čísel. Na druhom riadku je n medzerou oddelených celých čísel. Každé z týchto čísel reprezentuje nárast alebo pokles nadmorskej výšky v poradí, v akom sa nachádzali v Romanovom pláne.

[4] To je tá časť roku, kedy je úplne neproduktívny a chce sa mu iba spať a behať za syslicami.

Formát výstupu

Na prvý riadok vypíšte dĺžku najdlhšej podpostupnosti so súčtom 0. Potom vypíšte na ďalší riadok ešte dve čísla oddelené medzerou – poradové číslo prvého a posledného čísla tejto podpostupnosti v pôvodnej postupnosti. Prvý prvok postupnosti má poradové číslo 0.

Ak takáto podpostupnosť neexistuje, vypíšte na jediný riadok výstupu hodnotu -1.

Pri riešení v jazyku `C++`, `Pascal`, `Java` a podobných, si dávajte pozor na maximálne číslo, ktoré sú schopné celočíselné premenné uložiť. Súčet všetkých čísiel v postupnosti môže v poslednej sade tieto limity prekročiť. Odporúčame preto použiť 64-bitové premenné, v jazyku `C++` preto namiesto typu `int` použite typ `long long int`.

Riešení v `Pythone` sa takýto typ obmedzení netýka, pretože premenné v ňom majú neobmedzenú veľkosť.

Hodnotenie

Vaše riešenie bude otestované na 3 sadách vstupov. Za vyriešenie každej sady získate 5 bodov. Tieto sady sa líšia veľkosťou vstupných údajov. Nech n je dĺžka postupnosti a x je prvok tejto postupnosti, teda prevýšenie. Potom v jednotlivých sadách platí:

Číslo sady	1	2	3
maximálne n	100	50 000	100 000
maximálne $\|x\|$	1 000 000	10	1 000 000

Príklady

vstup

```
10
5 2 17 3 -3 4 6 -5 -5 50
```

výstup

```
6
3 8
```

Ak si pozrieme podpostupnosť $[3, -3]$, vidíme, že táto podpostupnosť má nulový súčet. Nie je však najdlhšia možná. Najdlhšia takáto postupnosť je postupnosť $[3, -3, 4, 6, -5, -5]$, ktorá má dĺžku 6. Táto postupnosť začína na pozícii 3 (číslované od 0) a končí na pozícii 8.

Takýto vstup by sa mohol nachádzať v ľubovoľnej sade.

vstup

```
5
200 30 60 -2 12
```

výstup

```
-1
```

V tomto vstupe neexistuje podpostupnosť so súčtom 0.

vstup

```
3
7 8 0
```

výstup

```
1
2 2
```

Riešením môže byť aj podpostupnosť dĺžky 1.