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 1. kola zimnej časti

Termín odoslania riešení tejto série je pondelok **7. decembra 2015**.

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 (http://oi.sk/archiv/2015/sl-2015-1-zad-B.pdf), úloha číslo 4.

Vrelo preto odporúčame riešiť aj Olympiádu v informatike. Nielenže sa naučíš nové veci, riešiš PRASK aj OI naraz, ale môžeš sa dostať aj na krajské kolo Olympiády.

1. Praktiky väznice Guantanámo

15 bodov za riešenie

Ak máte akékoľvek otázky ohľadom tejto úlohy napíšte Máriovi Lipovskému na mario@ksp.sk

Dvaja vedúci Prasku nedávno cestovali po svete a rozhodli sa, že navštívia väznicu Guantanámo. Šikovní väzni si ich hneď všimli, zobrali im oblečenie a vymenili sa s nimi. Pre stráže všetci vyzerajú rovnako a nikto našim vedúcim neuverí, že tam nepatria. To by predsa mohol povedať každý.

Vedúci si teda musia začať zvykať na väzenský život. Na dennom programe majú skoré budíčky, malé porcie jedla, studené sprchy, žiadny internet a guantanámske šarády. Veľmi sa im to tam nepáči, a tak sa rozhodli utiecť. Nebude to však také ľahké. Skúšali kopať tunel, preliezť plot aj podplatiť stráže, ale nič z toho sa im nepodarilo. Teraz majú nový nápad – utiecť počas obeda. Chcú pri tom využiť fakt, že na obed ich volajú v poradí, v akom sú napísaní vo veľkom zozname väzňov.

Každý väzeň dostal nové meno. Toto meno je zložené z prvých štyroch písmen abecedy a každé písmeno je použité práve raz. Všetky mená sú tiež rovnako dlhé. Takže mená, aké mohli dostať sú napríklad “ABCD” alebo “DCAB”, ale nemohli dostať mená “DAAB”, ani “FGAB”, ani “ABC”. Tieto mená sú zapísané v zozname väzňov v abecednom poradí (presne tak, ako by boli napísané v slovníku). Vedúci potrebujú zistiť viacero vecí pred tým, ako môžu utiecť. Po tom, čo kopali tunel sú ale dosť unavení. Pomôžete im?

Úloha

Okrem samotných odpovedí k úlohám popíšte aj postup, ako ste sa k výsledkom dopracovali.

a) (2 body) Vedúcich by zaujímalo, koľké v poradí je v zozname väzňov napísané meno “DBAC”.

b) (3 body) Vedúci sa boja, že ich presunú do najväčšej väznice na svete, kde používajú rovnaký systém mien, akurát každé meno má 15 písmen. Takže prvé v zozname je teraz meno “ABCDEFGHIJKLMNO”. Zistite, koľkým väzňom môže dať takáto väznica meno.

c) (1 bod) V Guantanáme nedávno prešli na 5-písmenové mená, lebo mali príliš veľa väzňov. Vedúci si všimli, že ich mená začínajú na písmeno D. „To je ale náhoda,” povedali si. Aby vedeli, aká veľká náhoda to naozaj je, máte im povedať, koľko väzňov dokopy môže mať meno začínajúce na D.

d) (1 bod) Pri úteku im bude pomáhať väzeň DABCE. Chcú vedieť, koľký v poradí je napísaný v zozname on.

e) (2 body) Počas obeda budú mať veľmi málo času na útek. DABCE bude utekať prvý a DEABC posledný. Spočítajte, koľko väzňov je medzi nimi v zozname.

f) (6 bodov) Útek sa nepodaril a Guantanámo sa rozhodlo zvýšiť bezpečnosť. Ako jedno z opatrení prešli na dlhšie mená a vedúcim by sa zišlo nájsť algoritmus, ktorý by im vedel povedať koľké v poradí je hocijaké meno zapísané v slovníku. Vymyslite a popíšte, ako funguje takýto algoritmus. Skúste aj dokázať,

že správne určí poradie každého mena. Ukážte, ako by fungoval na konkrétnych menách “FACDEB”, “AEGFDBCH”, “DIECAHFGB” a “BHACDEFGJI”.

Algoritmus je postup jednoduchých krokov, pomocou ktorých vyriešime problém – zistíme poradie mena v zozname. Algoritmus musí tiež zistiť správne poradie pre ľubovoľné meno, ktoré spĺňa všetky podmienky zadania – každé písmeno len raz a meno obsahuje prvé písmena abecedy.

Dajte si pozor na to, že v zozname sú vždy napísané len rovnako dlhé mená – teda ak rátame poradie slova “ABCDEF”, tak bude prvé, lebo nie je žiadne slovo dĺžky 6, ktoré spĺňa podmienky a ktoré by v abecede ležalo pred “ABCDEF”.

2. Praktické regulárne výrazy

15 bodov za riešenie

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

Každý z nás pri práci s počítačom občas hľadá nejaký text – či už je to na webstránke alebo v dokumente. Častokrát stačí, ak stlačíme `Ctrl+F`, napíšeme hľadaný výraz a program nám nájde všetky jeho výskyty. Ak však potrebujeme vyhľadať viacero synoným toho istého slova, všetky mená začínajúce veľkým písmenom a končiace na `„ka"` alebo platnú e-mailovú adresu, bežné nástroje prestávajú stačiť. Vtedy môžeme siahnuť po *regulárnych výrazoch*.

Regulárne výrazy patria medzi najpoužívanejšie nástroje na vyhľadávanie v texte. V tomto zadaní si popíšeme základy ich tvorby postačujúce na riešenie súťažných úloh. Pokojne si toho ale o regulárnych výrazoch prečítajte aj viac, takmer určite sa vám získané vedomosti ešte budú hodiť praxi.

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"`).[3]

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 seba, 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.

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

- Ak chceme, aby sa mohla časť vzorky v reťazci zopakovať, použijeme *kvantifikátor*. Základné kvantifikátory sú „`?`" (nulakrát alebo jedenkrát) a „`*`" (ľubovoľne veľakrát, vrátane 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) (3 body) Napíšte vzorku, ktorej zodpovedajú slovenské mobilné čísla patriace operátorovi Pomaranč. Slovenské telefónne čísla majú 10 cifier. Prvé dve sú čísla „`09`" nasledované ďalšou dvojicou, ktorá udáva operátora (do úvahy neberieme prenesené čísla). Sieť Pomaranč používa nasledovné dvojice: „`05`", „`06`", „`07`", „`08`", „`15`", „`16`", „`17`" a „`18`". Zvyšných 6 cifier môže byť ľubovoľných.

b) (2 body) Napíšte vzorku, ktorej zodpovedajú iba reťazce skladajúce sa z písmena „`a`" a dĺžka tohto reťazca je deliteľná tromi. To znamená, že má prijímať reťazce ako „`aaaaaa`" alebo „`"` (prázdny reťazec o dĺžke 0), ale nemá prijímať reťazce ako „`aaaa`".

c) (2 body) Napíšte vzorku, ktorej zodpovedajú iba reťazce skladajúce sa z písmena „`a`" a dĺžka tohto reťazca je nepárna.

d) (4 body) Napíšte vzorku, ktorej zodpovedajú iba reťazce zložené z písmen „`a`", „`b`" a „`c`", pričom všetky písmená „`a`" sa v reťazci nachádzajú pred všetkými písmenami „`b`".

e) (4 body) Napíšte vzorku, ktorej zodpovedajú iba reťazce zložené z písmen „`a`", „`b`" a „`c`" a žiadne dve za sebou idúce písmená reťazca **nie sú rovnaké**.

V riešení týchto úloh neodovzdaj iba hľadaný regulárny výraz, ale vysvetli, ako si našiel daný regulárny výraz a prečo tvojmu 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 "test 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.

3. Prísne tajné správy

15 bodov za riešenie

Ak máte akékoľvek otázky ohľadom tejto úlohy napíšte Edovi "Baklažánovi" Batmendijnovi na baklazan@ksp.sk

Ako agenti Príšerne Rýchlej Agentúry Stíhajúcej Kriminálnikov ste boli v rámci operácie "EVA" poverení získaním tajných informácií z komunikácie dvojice teroristov. Viete, že teroristi spolu komunikujú tak, že si posielajú obrázky, do ktorých rôznymi spôsobmi ukrývajú tajné správy. Podarilo sa vám odchytiť niekoľko takýchto obrázkov, ostáva už len nájsť v nich tajné správy.

Úloha

Zo stránky ksp.sk/~prask/specialne/2/1/3/tajne_spravy.zip si stiahnite súbor `tajne_spravy.zip`, v ktorom nájdete šesť podúloh. Každá podúloha pozostáva z jedného alebo dvoch obrázkov, v ktorých je ukrytá tajná správa. Tajná správa je vždy nejaký link. Keď na tento link pôjdete, budú vám automaticky pripočítané body. Prvé tri podúlohy sú každá za dva body, druhé tri podúlohy majú každá hodnotu tri body.

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. Pevná veža z kociek

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 Kubovi Havelkovi na kubo@ksp.sk

Istý Cézar veľmi rád hádzal kocky. Nevedel však, že rozloženie čísel na kocke by malo podliehať špecifickému dizajnu, preto boli čísla od 1 do 6 vytesané do kociek náhodne. Dokonca rôzne kocky ich mali usporiadané inak.

Kocky sú hodené. A čo teraz s nimi? Keďže bol Cézar veľký fanúšik bojovej techniky, zaumienil si postaviť z nich nový model bojovej veže. Strávil niekoľko hodín opatrným ukladaním kociek na seba a vylaďovaním otočenia kociek tak, aby boli pekne zarovnané a ani kúsok nevytŕčal von. Keď ju konečne dostaval, začal sa trochu obávať. Veža je tenká a niekto by ju mohol jediným drgnutím zbúrať. Treba si preto spraviť zálohu.

Začal teda spisovať všetky potrebné informácie. Najprv na prvý riadok napísal počet použitých kociek. Potom do ďalšieho riadku napísal číslo na vrchnej stene najvyššej kocky. Do každého z nasledujúcich riadkov napísal medzerami oddelené štyri čísla, ktoré boli vidieť na danej kocke z boku.

Úloha

Vašou úlohou je napísať a odovzdať program, ktorý vypočíta súčet čísel, ktoré sú na tých stenách kociek, ktoré nevidno, lebo sú zakryté podlahou alebo inou kockou.

Program bude testovaný na piatich sadách vstupov. Pre jednotlivé sady platia obmedzenia na výšku veže uvedené v tabuľke. Body dostanete za každú sadu, ktorú úspešne vyriešite.

Číslo sady	1	2	3	4	5
Maximálna výška veže	1	10	50	100	1000

Vstup

Vstup sa nachádza na štandardnom vstupe. Na jeho načítanie môžete použiť v Pascale funkciu `readln(premenna)`, v Pythone funkciu `premenna=input()` a v C++ funkciu `cin >> premenna`.

Vstup vyzerá tak, ako bol popísaný v zadaní. Na prvom riadku je číslo n udávajúce počet kociek a teda aj výšku veže. Na ďalšom riadku je jedno číslo, ktoré udáva počet bodiek na vrchnej stene najvyššej kocky. Nasleduje n riadkov, ktoré postupne (od vrchu po spodok) popisujú, ktoré čísla sú na bočných stranách všetkých kociek. V prvom takomto riadku sú teda štyri čísla, ktoré určujú čísla na bokoch najvyššej kocky, v druhom riadku štyri čísla z bokov kocky pod ňou atď.

Výstup

Výsledok, teda súčet zakrytých čísel, vypíšte na štandardný výstup, v Pascale na to slúži funkcia `writeln(vysledok)`, v Pythone `print(vysledok)` a v C++ `cout << vysledok`.

Nezabudnite za číslom dať koniec riadka. `writeln()` a `print()` to za vás spravia automaticky, v C++ musíte napísať ešte `cout << endl`.

Príklady

vstup

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

výstup

```
2
```

Jediná kocka, z ktorej nevidíme práve číslo 2.

vstup

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

výstup

```
18
```

Tu sú na seba poukladané tri kocky. Z najvrchnejšej nevidíme číslo 6, z druhej čísla 2 a 4 a z tretej čísla 1 a 5. To dáva v súčte 18.

5. Perfektné leukoplasty

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 odovdzaj pdf-ko s prezývkou, ktorú používaš na Liahni.*

Ak máte akékoľvek otázky ohľadom tejto úlohy napíšte Mišovi Štrbovi na faiface@ksp.sk

Žaba je perfekcionista. Veď sa len pozrite na jeho vlasy. Každý začína a končí presne tam, kde má, každý má svoje miesto a žiaden nie je navyše. Okrem toho je Žaba zdravotník[4]. A čo robí zdravotník? Lepí leukoplasty, ako keby sme nemali vlastné ruky. Na každú ranu dva, lebo jeden proste nestačí.

Keďže je Žaba perfekcionista, potrebuje presne vedieť, akú plochu na koži jeho pacienta pokrývajú práve nalepené leukoplasty. Práve sa zaoberá nejakými zložitými paralelnými algoritmami[5], na toto nemá čas a preto potrebuje aby ste mu s tým pomohli vy.

Úloha

Dostanete zadané pozície rohov dvoch leukoplastov. Musíte ich načítať zo vstupu. Máte napísať program, ktorý zistí a vypíše, koľko veľa (aký obsah) pacientovej kože je zalepenej. V prípade, že sa leukoplasty prekrývajú, oblasť, na ktorej sa prekrývajú zarátajte len raz.

Každý leukoplast je obdĺžnik v štvorcovej sieti, ktorého strany sú rovnobežné s osami x a y (skrátka, obyčajný rovný obdĺžnik).

Vstup

Vstup sa nachádza na štandardnom vstupe. Na jeho načítanie môžete použiť v Pascale funkciu `readln(premenna)`, v Pythone funkciu `premenna=input()` a v C++ funkciu `cin >> premenna`.

[4] Síce len pre dievčatá, ale to je teraz vedľajšie

[5] A popritom čaká, kým mu kuriér prinesie O(log n) počítačov...

Vstup sa skladá z dvoch riadkov, každý popisuje jeden leukoplast. Na prvom riadku sa nachádzajú súradnice ľavého dolného (dve čísla – x-ová a potom y-ová súradnica) a pravého horného rohu (ďalšie dve čísla – opäť najskôr x-ová a potom y-ová súradnica) prvého leukoplastu. Na druhom riadku sa nachádzajú súradnice rohov druhého leukoplastu v rovnakom formáte.

Všetky súradnice sú väčšie (alebo rovné) ako 0 a menšie ako 1 000.

Výstup

Výsledok, teda obsah kože zakrytej leukoplastami, vypíšte na štandardný výstup, v Pascale na to slúži funkcia `writeln(vysledok)`, v Pythone `print(vysledok)` a v C++ `cout << vysledok`.

Nezabudnite za číslom dať koniec riadka. `writeln()` a `print()` to za vás spravia automaticky, v C++ musíte napísať ešte `cout << endl`.

Príklady

vstup

```
1 1 8 4
8 4 14 8
```

výstup

```
45
```

Leukoplasty sa vôbec neprekrývajú, preto je výsledkom súčet ich obsahov.

vstup

```
0 0 20 20
5 5 15 15
```

výstup

```
400
```

V tomto prípade sa druhý leukoplast celý nachádza v prvom. Preto do celkového povrchu nepridáva žiadnu plochu. Obsah prvého leukoplastu je `400`.

vstup

```
1 1 5 12
3 11 13 14
```

výstup

```
72
```

Prvý obdĺžnik má obsah `44`, druhý má obsah `30`. Avšak prekrývajú sa na ploche, ktorá má obsah `2`. Preto celková plocha, ktorú pokrývajú je `44 + 30 - 2 = 72`.