Leták letnej 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 letnej časti

Termín odoslania riešení tejto série je pondelok **14. marca 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.

1. Perníková chalúpka

15 bodov za riešenie

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

Janko a Marienka sú väzňami v perníkovej chalúpke a radi by ušli. Marienka už aj vymyslela plán. Ježibaba totiž každý deň chodí spolu s Marienkou do lesa zbierať čarovné bylinky. V tom čase by Janko mohol ujsť a kým by sa Ježibaba s Marienkou vrátili, stihol by už byť dostatočne ďaleko a zavolať nejakú pomoc pre Marienku.

Celý plán však má jeden háčik. Marienka sa totiž s Jankom môže dohadovať iba večer (keď upratuje okolo chlievika) a Ježibaba sa až ráno rozhodne, kedy pôjde na bylinky (vtedy sa to dozvie aj Marienka). Ak by sa Janko pokúsil o útek, keď je Ježibaba v chalúpke, zle-nedobre by sa mu povodilo a Ježibaba by sprísnila opatrenia. Preto musí Marienka čas Ježibabinho odchodu do lesa Jankovi zakódovať do toho, ako mu naservíruje raňajky.

Úloha

Janko máva každý deň na raňajky niekoľko koláčov, z ktorých každý je buď makovník, alebo orechovník. Marienka mu ich vždy v kuchyni poukladá do jedného radu na podlhovastý podnos, pričom sa sama rozhodne, na ktorých pozíciách v rade budú makovníky a na ktorých orechovníky (Ježibaba má vždy v zásobe napečenú hŕbu makovníkov aj orechovníkov, takže ak by Marienka chcela, mohla by Jankovi naservírovať samé makovníky, alebo samé orechovníky). Následne Ježibaba podnos odnesie Jankovi. Keďže z jedného konca Ježibabinho podnosu je kúsok odštiepený, vie Janko jednoznačne určiť na ktorej strane je začiatok radu koláčov a na ktorej strane je jeho koniec.

a) (2 body) Janko máva na raňajky vždy 5 koláčov. Ježibaba chodieva na bylinky niekedy medzi 11:00 a 14:50. Marienka by Jankovi chcela zakódovať čas Ježibabinho odchodu s presnosťou na desať minút, teda chce vedieť zakódovať ľubovoľný z časov 11:00, 11:10, 11:20, ..., 14:50 (spolu 24 možností). Navrhnite spôsob, ako to urobiť. Presnejšie, navrhnite dve veci:

   - Postup, podľa ktorého sa Marienka na základe času Ježibabinho odchodu rozhodne, ktoré z koláčov majú byť makovníky a ktoré orechovníky.
   - Postup, podľa ktorého Janko na základe donesených koláčov zistí, v ktorom čase Ježibaba odíde.

   Nezabudnite, že Janko s Marienkou sa môžu večer na spôsobe kódovania dohodnúť, od rána sa však už rozprávať nemôžu.

b) (2 body) Janko máva na raňajky 8 koláčov. S akou veľkou presnosťou mu Marienka vie zakódovať čas Ježibabinho odchodu? Presnejšie povedané, koľko rôznych možností môže do ôsmich koláčov zakódovať? Svoju odpoveď zdôvodnite!

c) (1 bod) Ježibaba odchádza do lesa vždy medzi 10:00 a 16:59. Keďže každá minúta je drahá, Marienka by chcela čas jej odchodu zakódovať s presnosťou na minútu (chce teda vedieť rozlíšiť 420 rôznych možností). Koľko najmenej koláčov by Janko musel mať na raňajky, aby sa jej to mohlo podariť? Svoju odpoveď zdôvodnite!

d) (5 bodov) Ježibaba si vypestovala jeden veľmi nepríjemný zvyk: keď ide podnos s koláčmi odniesť Jankovi, najprv jeden z koláčov zje. Potom sa zapýri a na miesto zjedeného koláča doplní nejaký koláč zo zásoby – možno rovnakého druhu ako ten zjedený a možno opačného. Marienka chce Jankovi vedieť zakódovať 240 rôznych možností (časy medzi 11:00 a 14:59 s presnosťou na minútu) tak, aby:

- V prípade, že Ježibaba zjedený koláč nahradila rovnakým, Janko normálne zistil, kedy Ježibaba odíde a ušiel v správnom momente.
- V prípade, že Ježibaba zjedený koláč nahradila opačným, Janko zistil, že sa niečo pokazilo a útek radšej odložil na nasledujúci deň.

Opäť teda treba navrhnúť dva postupy, prvý pre Marienku, podľa ktorého na základe času Ježibabinho odchodu poukladá na podnos koláče a druhý pre Janka, podľa ktorého na základe koláčov na podnose buď zistí čas Ježibabinho odchodu, alebo aspoň zistí, že Ježibaba niektorý koláč vymenila za koláč opačného druhu. Váš spôsob kódovania musí fungovať vo všetkých prípadoch, bez ohľadu na to, ktorý koláč Ježibaba vymení.

Počet koláčov, ktoré Janko dostane na raňajky, si tento raz môže určiť Marienka. Tento počet však musí Ježibabe oznámiť ešte večer, nemôže sa teda o počte koláčov rozhodnúť až na základe času Ježibabinho odchodu. Snažte sa vymyslieť riešenie používajúce čo najmenej koláčov. Za ľubovoľné funkčné riešenie tejto podúlohy môžete získať až 3 body. Plných 5 bodov môžete dostať, ak bude vaše riešenie využívať iba 9 koláčov (áno, dá sa to :)).

e) (5 bodov) Ježibaba má rovnaký nepríjemný zvyk ako v podúlohe d), tentoraz však Janko s Marienkou navyše vedia, že sa chystá Janka upiecť už pozajtra ráno a teda si nemôžu dovoliť čakať ešte jeden deň. Preto sa musia dohodnúť na takom spôsobe kódovania, aby Janko vedel zistiť čas Ježibabinho odchodu, aj keby ktorýkoľvek koláč zjedla a nahradila koláčom opačného druhu. Čas Ježibabinho odchodu chcú tentoraz vedieť kódovať s presnosťou na sekundu (každá sekunda predsa môže byť rozhodujúca!), pričom Ježibaba môže ísť do lesa kedykoľvek medzi 9:00:00 a 20:59:59, musia teda vedieť rozlíšiť 43 200 rôznych možností.

Podobne ako v podúlohe d), aj v tejto podúlohe si môžete počet koláčov zvoliť sami, pričom opäť je cieľom použiť čo najmenej koláčov. Vaše riešenie však musí fungovať vždy, bez ohľadu na to, ktorý koláč Ježibaba vymení. Za akékoľvek funkčné riešenie môžete dostať až 3 body. Plných 5 bodov môžete dostať, ak vaše riešenie nepoužíva viac ako 34 koláčov.

f) (3 bonusové body) Bonusové body môžete získať, ak vyriešite podúlohu e) iba s pomocou 25 koláčov, ktoré však tentoraz Marienka nebude servírovať v jednom dlhom rade, ale v štvorci 5×5 na štvorcovom podnose (ktorý má v jednom rohu kúsok odbitý, takže Janko vie povedať, kde je ľavý horný roh :)).

2. Počítače zvnútra

15 bodov za riešenie

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

Zamysleli ste sa niekedy nad tým, ako funguje počítač? Na tej úplne najnižšej úrovni: tam, kde je to len hŕba súčiastok, ktoré sú medzi sebou rôzne poprepájané. Najprimitívnejšie súčiasky typicky rozlišujú iba dva možné stavy: ide mnou prúd a nejde mnou prúd, resp. som pod napätím a nie som pod napätím. Ako niečo takéto vôbec dokáže sčítať dve čísla? To si skúsime ukázať v tejto úlohe.

To, čo si ukážeme v tejto úlohe nie je úplne to isté, čo sa používa v počítačoch. Ale je to naozaj blízko. Najbližšie, ako vieme ísť bez toho, aby sme potrebovali fyziku. A môžem vás ubezpečiť, že týmito myšlienkami začínala éra počítačov :)

Hradlá

Počítač, ktorý budeme stavať, bude poznať iba dve hodnoty – hodnotu 0 a hodnotu 1. Skladať sa bude z káblov a logických súčiastok, ktoré budeme nazývať hradlá. Hradlá sú krabičky, ktoré majú niekoľko vstupov a niekoľko výstupov[3]. Keď na každý vstup privedieme hodnotu 0 alebo 1, hradlo ich vo vnútri spracuje a na každý výstup dá určenú hodnotu, ktorú vypočítalo. Väčšinou hradlá počítajú nejaké logické funkcie.

Hradlo NOT – je jedným zo základných hradiel, má jeden vstup a jeden výstup. Na výstup vracia hodnotu opačnú k tomu, čo dostalo na vstupe. Ak teda hradlu *NOT* dáme na vstup 0, vráti nám 1 a naopak, ak mu dáme na vstup 1, vráti nám 0.

Hradlo AND – má dva vstupy a jeden výstup. Na výstup vráti 1 iba vtedy, ak boli obe hodnoty na vstupe 1. V opačnom prípade (aspoň na jednom vstupe bola hodnota 0) na výstup vráti 0. Toto samozrejme zodpovedá logickému *AND* (*a zároveň*), ktoré poznáte z matematiky.

Hradlá môžeme medzi sebou rôzne skladať a prepájať pomocou káblikov. Vieme teda napojiť výstup jedného hradla na vstup iného hradla a takýmto spôsobom získať zložitejšie obvody. Takisto môžeme káble rozvetviť, čo

[3] To, na akom fyzikálnom princípe tieto krabičky fungujú vnútri, sa v priebehu histórie menilo. Najprv sa používali relé, neskôr elektrónky. V súčasných počítačoch sa používajú tranzistory.

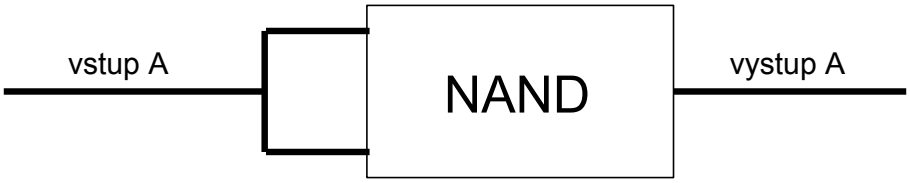
spôsobí, že tá istá hodnota pôjde do viacerých vstupov rôznych alebo rovnakých hradiel. Vhodným skombinovaním viacerých hradiel môžeme dostať obvod, ktorý sa celý správa ako nejaké (iné) hradlo.

Čo by sme však nemali s hradlami robiť, je spájať dva výstupy do jedného káblika, zaviesť výstup hradla, alebo čokoľvek čo sa z tohto výstupu počíta, na vstup toho istého hradla. Takéto konštrukcie totiž nebudú fungovať.

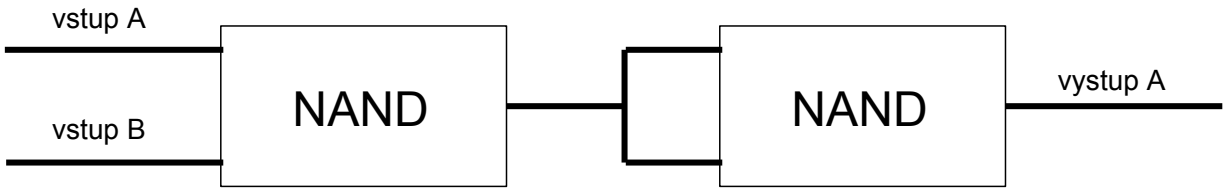
Príklad

Predstavme si, že máme iba zásobu hradiel *NAND*. **Hradlo NAND** má dva vstupy a jeden výstup. Hradlo vráti na výstup hodnotu 0 iba ak sú obe vstupné hodnoty rovné 1, vo všetkých ostatných prípadoch vráti na výstup 1. Iba pomocou hradiel tohto typu by sme chceli poskladať hradlá *NOT* a *AND*.

Hradlo *NOT* má iba jeden vstup, ktorého hodnotu máme obrátiť. Hradlo *NAND* ale potrebuje dva vstupy. Vstup, ktorý dostaneme, preto rozvetvíme na dva a zapojíme do *NAND*. Výstup z tohoto *NAND*-u bude výstupom nášho obvodu. Výsledná hodnota bude naozaj opačná k tomu, čo sme mali na začiatku. Ak sme totiž na vstupe obvodu mali 1, na hradlo *NAND* sa dostanú dve jednotky, teda toto hradlo vráti 0. Ak sme na vstupe mali 0, na *NAND* sa dostanú dve nuly, teda vráti nulu. Schému nášho obvodu si môžete pozrieť na obrázku.



Ešte chceme spraviť hradlo *AND*. Uvedomme si, že *NAND* vracia presne opačnú hodnotu k tomu, čo vracia *AND*. Z hodnôt na vstupe teda môžeme vypočítať *NAND* a tento výsledok zmeniť pomocou postupu, ktorý sme si popísali vyššie (v podstate použijeme hradlo *NOT*). Schému si môžete pozrieť nižšie.



Úloha

Vašou úlohou bude vytvoriť niekoľko zložitejších logických hradiel. Úlohy na seba nadväzujú a na riešenie niektorých podúloh sa vám zídu hradlá z predchádzajúcich podúloh. Ak vám to zadanie povoľuje, tak môžete používať hradlá z iných podúloh ako hotovú krabičku s daným názvom, ktorá sa správa tak, ako má. Pritom nemusíte mať príslušnú podúlohu ani vyriešenú. Hradiel každého typu môžete použiť **ľubovoľne veľa**.

a) (2 body) **Hradlo OR** – vašou úlohou je vytvoriť hradlo, ktoré má dva vstupy a jeden výstup. Na výstup vráti 1, ak je aspoň jedna zo vstupných hodnôt 1. 0 vráti, ak sú obe vstupné hodnoty 0. Môžete použiť **iba** hradlá *NOT* a *AND*.

b) (2 body) **Hradlo SORT** – vytvorte hradlo s dvoma vstupmi a dvoma výstupmi. Toto hradlo usporiada vstupné hodnoty, to znamená, že na výstup označený *výstup A* vráti menšiu z dvoch hodnôt a na výstup označený *výstup B* tú väčšiu. Ak sú hodnoty na vstupe rovnaké, vráti ich na výstupy v ľubovoľnom poradí (však sú rovnaké). Môžete použiť hradlá *NOT*, *AND* a *OR*.

c) (2 body) **Hradlo IMP** – vytvorte hradlo s dvoma vstupmi a jedným výstupom. Na výstup vráti toto hradlo 0, iba ak je prvý vstup (označený *vstup A*) rovný 1 a druhý vstup (*vstup B*) rovný 0. V opačnom prípade vráti na výstup hodnotu 1. Toto hradlo teda zodpovedá matematickej implikácii. Môžete použiť hradlá *NOT*, *AND* a *OR*.

Ako sme si povedali na začiatku, našim cieľom bude sčítavať čísla. Poznáme však iba dve cifry 0 a 1, preto budeme všetky čísla zapisovať iba pomocou týchto dvoch cifier – v *binárnej* (dvojkovej) sústave. Binárna sústava

je vec celkom zaujímavá a ak jej chcete poriadne rozumieť, odporúčame vám prečítať si o nej niečo na internete[4].

Na vyriešenie tejto úlohy vám však stačí vedieť, ako čísla v binárnej sústave sčítavať. Ak chceme sčítať dve binárne čísla, môžeme si ich napísať jedno pod druhé a sčitovať zprava doľava po cifrách (presne ako sa to robí pri číslach v desiatkovej sústave). Platí, že $0+0=0$, $0+1=1$ a $1+0=1$. Jediný rozdiel je pri $1+1$, lebo to sa nemôže rovnať 2, keďže takú cifru nepoznáme, ale rovná sa 0 s tým, že jedna 1 sa prenesie do ďalšieho rádu, teda k nasledujúcej cifre (podobne, ako keď pri sčítaní dvoch čísel v desiatkovej sústave dostaneme súčet dvoch cifier väčší ako 9).

Nižšie vidíte tri príklady toho, ako funguje sčítanie binárnych čísel. V prvom príklade nastáva prenos pri sčítavaní druhej, štvrtej, piatej a šiestej cifry od konca. Všimnite si, že pri sčítavaní šiestej cifry sčítavame dokopy až tri 1 (po jednej z každého čísla a jednu z prenosu), čoho výsledkom je 1 a presunutá 1 do vyššieho rádu.

```
  00101011          00000001          00011011
+ 10111010        + 01111111        + 11111001
----------        ----------        ----------
  11100101          10000000         100010000
```

d) (3 body) **Hradlo XORc** – vytvorte hradlo s dvoma vstupmi a dvoma výstupmi. Jeho úlohou bude sčítať hodnoty, ktoré dostane na vstupe. Na výstup *vystup A* vráťte výsledok tohto sčítania (samozrejme, namiesto hodnoty 2 vraciate hodnotu 0) a na výstup *vystup B* vráťte 1, ak sa prenáša jednotka do vyššieho rádu, inak naň vráťte 0. Použiť môžete hradlá *NOT*, *AND* a *OR*.

e) (2 body) **Hradlo XOR3c** – vytvorte hradlo s tromi vstupmi a dvoma výstupmi, ktoré sčíta tri hodnoty, ktoré má na vstupe a na *vystup A* vráti výsledok tohto sčítania (namiesto výsledku 2 vráti 0 a namiesto výsledku 3 vráti 1) a na *vystup B* dá 1, ak sa pri sčítaní prenášala jednotka do vyššieho rádu. Môžete použiť hradlá *NOT*, *AND*, *OR* a *XORc*.

f) (4 body) **Hradlo SUM** – cieľom bude vytvoriť hradlo, ktoré sčíta dve osemciferné binárne čísla x a y, ktoré môžu začínať prebytočnými nulami. Cifry týchto čísiel si označíme $x_0, x_1 ... x_7$ a $y_0, y_1 ... y_7$, pričom x_0 a y_0 sú najmenej významné cifry (tie úplne napravo). Týmito názvami si tiež označíme 16 vstupov, ktoré bude naše hradlo mať. Výstupov bude mať 9 a budú označené z_0 až z_8, a dokopy majú tvoriť číslo $x+y=z$, pričom z_0 je najmenej významná cifra čísla z. Popíšte, ako takéto hradlo vytvoriť. Na riešenie môžete použiť hradlá *NOT*, *AND*, *OR*, *XORc* a *XOR3c*.

Odovzdávanie a hodnotenie

Táto úloha sa dá odovzdávať dvomi spôsobmi. Prvým z nich je odovzdať klasické .pdf, v ktorom nakreslíte hradlá podobne ako v časti *Príklad* a popíšete, ako ste prišli na svoje riešenie. Za postup, ktorým ste sa snažili riešiť danú podúlohu, môžete získať čiastočné body aj v prípade, že vaše riešenie úplne nefunguje.

Druhá možnosť je stiahnuť si program *Logisim* (je to program špeciálne určený na navrhovanie logických obvodov), vaše riešenia nakresliť v ňom a odovzdať ich na automatické testovanie. To funguje podobne ako pri programátorských úlohách: testovač vám hneď povie, či sú vaše riešenia správne, alebo nie a pridelí vám príslušný počet bodov. V prípade že vaše riešenie nefunguje, môžete si ho skúsiť opraviť a odovzdať znova (a toto môžete opakovať ľubovoľne veľa krát). Ak niektoré podúlohy (najlepšie všetky) vyriešite takýmto spôsobom, nemusíte k nim už odovzdávať žiaden slovný popis. Technické detaily k tomuto spôsobu odovzdávania nájdete na tejto stránke: https://prask.ksp.sk/logisim/.

Praktická úloha

Už v staroveku ľudia šifrovali správy, v ktorých boli citlivé informácie. V tejto úlohe si vyskúšate lámanie niektorých jednoduchých šifier.

3. Písanie si v tajnosti

0 bodov za riešenie

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

Andrej si minule prečítal knižku o historických šifrách a niektoré z nich sa mu veľmi zapáčili. Preto ich hneď naučil aj svojho kamaráta Adama a začali si posielať zašifrované odkazy. Teraz sa všade chvastajú, že nikto nikdy nezistí čo si píšu. Preto sme zachytili viacero ich správ a rozhodli sme, sa že ich dešifrujeme.

[4] Napríklad tu: [http://www.gym-informatika.estranky.cz/clanky/dvojkova-sustava.html] (http://www.gym-informatika.estranky.cz/clanky/dvojkova-sustava.html)

Úloha

Za úlohu máte dešifrovať niekoľko správ – odchytené správy Adama a Andreja.

Vo všetkých úlohach platí, že budú písať len anglickou 26 písmenovou abecedou a šifrovať budú iba písmena. To znamená, že všetky špeciálne znaky ako bodky, čiarky, úvodzovky alebo dokonca čísla zostanú nezmenené. Navyše, nerozlišujú medzi veľkosťou písmen a do výsledného zašifrovaného textu píšu iba veľké písmená.

a) (3 body) V prvej podúlohe viete, že Andrej poslal Adamovi správu zašifrovanú pomocou Cézarovej šifry. Prebiehalo to tak, že sa najprv dohodli na nejakom kladnom celom čísle, ktoré budeme nazývať kľúč k. Abecedu si napísali na kružnicu, a teraz vždy keď Andrej posiela Adamovi list, každé písmeno v tomto liste nahradí písmenom, ktoré leží na tejto kružnici o k pozícií ďalej. Na kružnicu si ju napísali preto, aby po písmene 'Z' nasledovalo opäť písmeno 'A'. Keď chce Andrej zašifrovať slovo `„zima"` s kľúčom $k = 2$, napíše `„BKOC"`.

Správa, ktorú sme odchytili sa nachádza tu: ksp.sk/~prask/2/3/3/sifraA.html

b) (5 bodov) Andrej sa rozhodol zvýšiť bezpečnosť, pretože Cézarova šifra bola veľmi ľahko rozlúštiteľná. Rozhodol sa použiť substitučnú šifru. Andrej si do prvého riadku napísal pekne v poradí všetky písmená abecedy. Do druhého riadku tiež napísal písmená abecedy, ale v nejakom inom poradí. Vyzerať to mohlo napríklad takto:

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
T	J	M	G	D	N	A	C	O	B	E	L	W	V	X	K	Z	Y	Q	R	P	I	H	U	F	S

Túto dvojicu riadkov si nazvime kľúč. Keď chce Andrej niečo zašifrovať a poslať Adamovi, každé písmeno v správe nahradí písmenom, ktoré je v kľúči napísané pod ním. Teda slovo `„Adam"` by zašifroval na `„TGTW"`. O niektorých častiach Andrejovej správy si však vieme tipnúť, čo reprezentovali v originálnej správe a tie nám pomôžu šifru postupne rozšifrovať.

Zašifrovaná správa sa nachádza tu: ksp.sk/~prask/2/3/3/sifraB.html

c) (7 bodov) Adam sa už naštval, že mu niekto stále číta správy, a teda sa rozhodol že to poriadne sťaží. Opäť používajú substitučnú šifru, ale tento raz si dal Adam pozor, aby v správe neboli žiadne ľahko uhádnuteľné časti ako pozdrav, podpis a podobne. Máte ale šťastie že správa je veľmi dlhá, to sa možno bude dať nejako využiť.

Správa sa nachádza tu: ksp.sk/~prask/2/3/3/sifraC.html

Odovzdávanie

Každý zašifrovaný text v sebe ukrýva tajné heslo. Toto heslo nie je zmysluplné, je to len zhluk písmen. Ak sa vám toto heslo podarí zistiť, choďte na stránku `„prask.ksp.sk/specialne/prask/2/3/3/heslo/"` (vraj lomítko na konci je dôležité) kde namiesto slova `„heslo"` napíšete zistené heslo. Ak ho budete mať rozšifrované správne, body sa vám pripíšu automaticky.

Programátorské úlohy

Tieto úlohy sú zamerané na praktickú tvorbu programov v niektorom vyššom programovacom jazyku ako je napríklad Python, C++ alebo Pascal. Na stránke odovzdávaš **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. Toto môžeš urobiť **s každou sadou najviac raz**.

Pre druhú sériu sú na Liahni ďalšie dve sady, ktorými si budeš môcť nahradiť programátorské úlohy z nej.

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. Zjedená pizza

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 Michalovi 'Žabovi' Anderlemu na zaba@ksp.sk

KSPáci majú radi jedlo – to vie hádam každý. A tak si počas posledných príprav pred sústredkom objednali do T2 pizzu, aby mali dosť síl na pobalenie kufrov. Keď odchádzali na sústredko, každý niečo niesol, ale krabice od pizze ostali tam – za ten týždeň, čo tu nebudeme, ich určite niekto vyhodí...

Sústredko prebehlo a všetci sa plní zážitkov vrátili do T2, kde s hrôzou zistili, že krabice od pizze sú stále tam. Adam si všimol, že krabica od jeho veľkej pizze má dvakrát takú dlhú stranu ako krabice od malých pízz a vložil 4 menšie krabice vedľa seba do jednej veľkej. Takéto šetrenie miesta sa mu zapáčilo. Zobral si do ruky meter a začal merať veľkosti všetkých krabíc. S prekvapením zistil, že dĺžka strany každej krabice bola nejaká mocnina dvojky. Do najväčšej krabice povkladal teda tak veľa menších, koľko sa dalo. Potom rovnako pokračoval so zvyškom. Zistite, koľko krabíc musel nakoniec Adam odniesť do kontajnera.

Úloha

Máte zadané dĺžky strán všetkých krabíc. Každá krabica je štvorcová a dĺžka jej strany je mocnina dvojky.

Do každej krabice vieme vkladať len menšie krabice. Do krabice s dĺžkou strany 2^k vieme vložiť vedľa seba najviac 4 krabice s dĺžkou strany 2^{k-1}. Ak zostáva v krabici voľné miesto, môžeme ho vyplniť aj menšími krabicami. Jednoducho, celková plocha krabíc, ktoré sú uložené vedľa seba, nemôže byť väčšia ako plocha vonkajšej krabice.

Vašou úlohou je zistiť, koľko krabíc ostane po tom, ako ich optimálne povkladáme do seba.

Mocniny 2

Mocniny čísla 2 sú postupne čísla 1, 2, 4, 8, 16, 32 ... Každú ďalšiu mocninu dostaneme, ak predchádzajúcu vynásobíme dvojkou. Zápis 2^k predstavuje číslo, ktoré získame, ak číslo 1 k-krát vynásobíme číslom 2. Napríklad $2^3 = 8$ alebo $2^0 = 1$.

Formát vstupu

V jednom riadku je 20 čísel $a_0, a_1, \ldots, a_{19}$ oddelených medzerami ($0 \leq a_i \leq 1\,000\,000$). Číslo a_i hovorí o tom, koľko máme krabíc s dĺžkou strany 2^i.

Na pamätanie si veľkých čísel[5] môžete použiť 64-bitové premenné typu `long long` v C++, `Int64` v Pascale.

Formát výstupu

Vypíšte jedno číslo – počet krabíc ktoré ostanú na konci.

Príklad

vstup	výstup
`0 0 5 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0`	`2`

Ostane krabica so stranou dlhou 8 (v ktorej sú 4 krabice s dĺžkou 4), a jedna krabica s dĺžkou 4, ktorá sa tam už nezmestila.

vstup	výstup
`0 0 13 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0`	`1`

[5] Približne v rozsahu -2^{63} až 2^{63}.

Balíme trinásť krabíc s dĺžkou strany 4*, jednu s dĺžkou* 8 *a jednu s dĺžkou* 16*. Môžeme ich pobaliť nasledovne: Najprv do krabice s dĺžkou strany* 8 *zabalíme jednu štvorkovú. Potom sa krabica* 8×8 *a 12 krabíc* 4×4 *presne vojdú do krabice* 16×16*.*

vstup	výstup
0 0 20 4 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0	5

Zostanú 4 najmenšie krabice a 1 najväčšia.

5. Zázračné karty

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 Michalovi 'Žabovi' Anderlemu na zaba@ksp.sk

Žaba má zázračné karty. Aspoň to tak tvrdí. V skutočnosti má úplne obyčajný balíček kariet, s ktorými robí všakovaké triky. Minule stretol Kozzu a jeden takýto trik mu predviedol. Karty premiešal, balíček párkrát prevrátil hore nohami, znovu premiešal... A na koniec Kozzu ohúril tým, že vedel presne povedať poradie, v akom karty sú. Kozza bol úplne nadšený. Hneď chcel, aby ho Žaba trik s kartami naučil. Žaba mu teda prezradil tajomstvo svojho úspechu: totiž, že na začiatku vedel, v akom poradí karty sú a keď ich miešal, vždy len zobral kartu zvrchu a dal ju na spodok, alebo zobral kartu zospodu a dal ju navrch, alebo celú kopu otočil hore nohami. Ako ale vedel tak rýchlo povedať, ako bude vyzerať finálne poradie? To si teraz naprogramujete!

Úloha

Každá karta je označená nejakým (nie nutne jedinečným) prirodzeným číslom. Máte zadaných n kariet, v poradí, v akom boli na začiatku v kope kariet, od spodku po vrch. Ďalej máme zadanú postupnosť q krokov: **D** – zoberieme kartu z vrchu kopy a presunieme ju **d**ole, **H** – zoberieme kartu zospodu kopy a presunieme ju **h**ore, a **R** – otočenie kopy (**r**everz). Pre dané poradie kariet na začiatku a postupnosť krokov zistite, v akom poradí budú karty na konci.

Formát vstupu

Na prvom riadku vstupu sú čísla n ($1 \leq n \leq 1\,000\,000$) a q ($1 \leq q \leq 1\,000\,000$) oddelené medzerou – počet kariet v kope a počet krokov miešania. Na ďalšom riadku je n medzerami oddelených čísel – čísla kariet v poradí, v akom boli na začiatku v kope od spodku po vrch. Čísla nepresiahnu 1 000 000 a môžu sa opakovať. Nasleduje riadok s q znakmi oddelenými medzerami. Každý znak popisuje jeden krok miešania – D (presun dole), H (presun hore) a R (reverz kopy).

Formát výstupu

Na výstup vypíšte n čísel – čísla kariet v poradí, v akom budú karty na konci triku v kope, odspodu po vrch.

Príklad

vstup	výstup
5 4 7 8 9 1 2 D D H R	1 9 8 7 2

Vykonávaním jednotlivých krokov sa bude poradie kariet v kope meniť nasledovne: 78912 → 27891 → 12789 → 27891 → 19872

vstup	výstup
3 7 1 2 3 R D D R H H R	1 3 2