MATEMATICKÁ OLYMPIÁDA NA STREDNÝCH ŠKOLÁCH

54. ročník, školský rok 2004/2005

Zadania úloh 1. kola kategórie P

Matematická olympiáda je súťaž pre študentov stredných škôl našej republiky. **Kategória P** je zameraná na programovanie a je určená študentom všetkých ročníkov.

Organizácia súťaže v kategórii P

Kategória P matematickej olympiády má tri postupové kolá — domáce, krajské a celoštátne.

V **I. kole** účastníci riešia štyri úlohy, uvedené v tomto letáku. Riešenia odovzdajú svojmu učiteľovi informatiky do **15. novembra 2004**. Učitelia informatiky odošlú riešenia v tomto termíne zo školy na príslušnú adresu podľa krajov takto:

Košický, Prešovský:
RNDr. G. Andrejková, CSc., KMI PF UPJŠ, Jesenná 5, 041 54 Košice
Žilinský:
RNDr. P. Varša, KI FRI Žilinská univerzita, Moyzesova 20, 010 26 Žilina
Banskobystrický:
Mgr. L. Huraj, KI FPV UMB, Tajovského 40, 974 01 Banská Bystrica
Trnavský, Trenčiansky, Nitriansky:
Doc. Ing. V. Štoffová, CSc., KI FPV UKF, tr. A. Hlinku 1, 949 74 Nitra
Bratislavský:
RNDr. A. Blaho, KVI FMFI UK, Mlynská dolina, 842 48 Bratislava 4

Najúspešnejší riešitelia domáceho kola sú pozvaní do **II. kola** (krajského), kde riešia štyri teoretické úlohy. Do **III. kola** (celoštátneho) sú pozývaní najúspešnejší riešitelia všetkých krajských kôl, pričom riešia tri teoretické úlohy a dve praktické úlohy pri počítači. Z najlepších riešiteľov tohto kola SK MO vyberie družstvá pre Medzinárodnú informatickú olympiádu a Stredoeurópsku informatickú olympiádu.

Predbežné termíny 54. ročníka MO, kategória P

I.	Domáce kolo	15. novembra 2004
II.	Krajské kolo	11. januára 2005
III.	Celoštátne kolo	6.-9. apríl 2005

Usporiadateľ súťaže

Matematickú olympiádu vyhlasuje *Ministerstvo školstva SR* v spolupráci s *Jednotou slovenských matematikov a fyzikov* a *Slovenskou komisiou Matematickej olympiády.* Súťaž organizuje *Slovenská komisia MO* a v jednotlivých krajoch ju riadia *krajské komisie MO* pri pobočkách JSMF. Na jednotlivých školách ju zaisťujú učitelia matematiky a informatiky. Celoštátne kolo MO, tlač materiálov a ich distribúciu po organizačnej stránke zabezpečuje IUVENTA v tesnej súčinnosti so Slovenskou komisiou MO.

Formálna úprava riešenia

Riešenia súťažných úloh domáceho kola kategórie P pozostávajú z dvoch častí:

Popis riešenia. Riešenia musia obsahovať podrobný popis použitého algoritmu, **zdôvodnenie jeho správnosti** a diskusiu o efektivite zvoleného riešenia (t.j. posúdenie časových a pamäťových nárokov programu). Algoritmus by mal byť jasný už z popisu riešenia bez toho, aby bolo potrebné nahliadnuť do programu.

Program. V úlohách **P–I–1**, **P–I–2** a **P–I–3** je potrebné k riešeniu pripojiť odladený program napísaný v jazyku Pascal, C alebo C++. (Musí sa dať skompilovať kompilátorom *FreePascal*, resp. *gcc*.) Program sa odovzdáva v písomnej forme (jeho výpis je teda súčasťou riešenia) i na diskete, aby bolo možné otestovať jeho funkčnosť.

Súbory na diskete pomenujte `p1`x`.pas/.c/.cpp`, kde x je číslo súťažnej úlohy. Disketu označte menom riešiteľa. Z jednej školy možno poslať všetky riešenia na jednej diskete. V tomto prípade pre každého riešiteľa vytvorte podadresár označený jeho priezviskom a disketu označte adresou školy.

V úlohe **P–I–4** je potrebné uviesť program pre ALIK.

Písomnú časť riešenia vypracujte čitateľne na listy formátu A4. **Každú úlohu začnite na novom liste** a v záhlaví uveďte vaše meno, triedu, adresu školy a označenie príkladu podľa tohto letáku. Zadania úloh nemusíte opisovať. Ak sa vám riešenie nezmestí na jeden list, uveďte na ďalších listoch vľavo hore svoje meno a označenie úlohy a očíslujte strany.

P-I-1

Kleofáš sa rozhodol, že začne vo veľkom podnikať a otvorí si veľkoprá-čovňu. Každý, kto príde, si bude môcť za pár korún prenajať práčku (ak bude nejaká voľná), vyperie si v nej svoje prádlo a spokojne sa vráti domov. Malá anketa medzi priateľmi ukázala, že záujem o takúto práčovňu by bol vskutku veľký. Čoskoro chudák Kleofáš úplne stratil predstavu, koľko vlastne práčok potrebuje kúpiť, aby vystačili pre všetkých zákazníkov. A preto sa obrátil na vás s prosbou o pomoc.

Úloha: Na vstupe je počet zákazníkov, ktorých Kleofáš očakáva počas jedného dňa. O každom vie, kedy príde a na ako dlho si chce práčku prenajať. Jednotliví zákazníci nie sú uvedení v žiadnom konkrétnom poradí.

Vašou úlohou je zistiť, koľko najmenej práčok Kleofáš potrebuje mať, aby si každý zákazník mohol v okamihu príchodu prenajať jednu práčku na požadovanú dobu. Okrem zistenia potrebného počtu práčok by mal váš program aj vytvoriť zoznam, podľa ktorého bude Kleofáš prideľovať práčky prichádzajúcim zákazníkom.

Formát vstupu: Prvý riadok vstupného súboru `pracky.in` obsahuje prirodzené číslo $N \leq 10\,000$ – počet zákazníkov. Na každom z nasledujúcich N riadkov sú informácie o jednom zákazníkovi: čas t_i, kedy príde a doba d_i, na ktorú si chce prenajať práčku. Môžete predpokladať, že t_i a d_i sú celé čísla od 1 do 1 000 000 000. Zákazníkom priradíme čísla od 1 do N v poradí, v akom sú uvedení na vstupe.

Formát výstupu: Prvý riadok výstupného súboru `pracky.out` má obsahovať jediné číslo P – najmenší počet práčok, ktoré Kleofášovi stačia. Tieto práčky očíslujeme číslami od 1 do P. Na nasledujúcich N riadkoch výstupného súboru majú byť čísla $a_1, \ldots, a_N$, kde a_i je číslo práčky, ktorú dostane pridelenú i-ty zákazník.

Príklad:

pracky.in	pracky.out
4	3
1000 1000	2
1900 900	1
1500 700	3
2000 500	2

P-I-2

Na nemenovanom internáte onedlho prebehnú každoročné preteky švábov. Preteky prebiehajú na prekážkovej dráhe, ktorá obsahuje zákernosti ako krájač na vajíčka a misku cukru. Šváby sú na trať vypúšťané v minútových intervaloch. Aby sa ich dalo rozoznať, každý dostane na chrbát kartičku s časom, kedy štartoval (teda prvý šváb dostane číslo 0, druhý 1, atď.). Organizátorov pretekov by zaujímalo, ako veľmi sa im šváby počas cvičného preteku zamiešali, aby vedeli, či netreba zväčšiť interval medzi štartujúcimi.

Rozhodli sa preto, že spočítajú počet takých dvojíc švábov, ktoré dobehli v opačnom poradí ako vybehli (t.j. šváb s vyšším číslom dobehol skôr ako šváb s nižším číslom). Ale keďže švábov je na dotyčnom internáte neuveriteľne veľa, potrebujú pomoc počítača.

Úloha: Váš program dostane na vstupe počet švábov N a poradie, v ktorom dobehli (t.j. nejakú permutáciu čísel od 0 do $N-1$). Na výstup má vypísať počet dvojíc, ktoré si vymenili poradie.

Formát vstupu: Prvý riadok vstupného súboru `preteky.in` obsahuje prirodzené číslo N $(1 \leq N \leq 30\,000)$ – počet švábov. Na druhom riadku je uvedených N navzájom rôznych čísel od 0 do $N-1$, medzi každými dvoma číslami je práve jedna medzera.

Formát výstupu: Výstupný súbor `preteky.out` má obsahovať jediný riadok a na ňom jediné číslo – počet dvojíc švábov, ktoré dobehli v opačnom poradí ako vybehli.

Príklad:

preteky.in	preteky.out
5 1 0 4 2 3	3

P-I-3

Fylogenetika je odbor biológie študujúci vývojové vzťahy medzi organizmami. Často používanou metódou je porovnávanie genómov. V tejto úlohe sa budeme zaoberať veľmi zjednodušenou verziou tohto problému.

Genóm budeme mať uložený ako reťazec pozostávajúci z písmen '`A`', '`C`', '`G`' a '`T`'. Budeme predpokladať, že vývoj druhu prebieha tak, že na začiatok alebo na koniec genómu sa pridajú nové gény – to je samozrejme veľké zjednodušenie (rozumej: úplný nezmysel). Na vstupe dostanete genómy niekoľkých organizmov a úlohou je nájsť medzi nimi všetky dvojice predok – potomok, t.j. také, že genóm predka je súvislým podreťazcom genómu potomka.

Formát vstupu: Vstupný textový súbor `fylogen.in` obsahuje niekoľko reťazcov zložených z písmen '`A`', '`C`', '`G`' a '`T`', reprezentujúcich genómy jednotlivých organizmov. Organizmy sú očíslované $1, 2, \ldots, n$; na i-tom riadku sa nachádza genóm i-teho organizmu. Môžete predpokladať, že reťazcov je najviac 50, každý z nich má najviac 50 znakov a žiadne dva reťazce nie sú rovnaké.

Formát výstupu: Výstupný textový súbor `fylogen.out` obsahuje zoznam všetkých dvojíc predok – potomok. Každý riadok súboru popisuje jednu z týchto dvojíc a pozostáva z čísla predka nasledovaného číslom potomka. Dvojice môžu byť vypísané v ľubovoľnom poradí, nesmú sa však opakovať.

Príklad:

fylogen.in	fylogen.out
ATAT	1 2
CATATG	1 3
CATATGA	1 4
CATATGG	2 3
	2 4

P-I-4

Študijný text – ALIK

Aritmeticko-logická integerová kalkulačka (skratka ALIK) je výpočtové zariadenie pracujúce s W-bitovými celými číslami v rozsahu 0 až $2^W - 1$ vrátane; v ďalšom texte pod *číslami* rozumieme vždy takéto čísla. Budeme ich obvykle zapisovať v dvojkovej sústave hrubým písmom a na začiatok dvojkového zápisu vždy doplníme príslušný počet núl, aby počet číslic (bitov) bol presne W. Väčšinou tiež nebudeme rozlišovať mezi číslom a jeho dvojkovým zápisom, takže i-tym bitom čísla budeme rozumieť i-ty bit jeho dvojkového zápisu (bity číslujeme zprava doľava od 0 po $W - 1$).

Pamäť stroja tvorí 26 *registrov* pomenovaných a až z. Každý register vždy obsahuje jedno číslo.

ALIK se riadi programom, čo je postupnosť priraďovacích príkazov typu *register* := *výraz*, kde *výraz* môže obsahovať konštanty (čísla zapísané v dvojkovej sústave), registre, zátvorky a nasledujúce operátory (grécke písmená označujú podvýrazy):

- $\boxed{\alpha + \beta \text{ (priorita 4)}}$ sčíta čísla α a β. Ak je výsledok viac ako $2^W - 1$, číslice vyšších rádov odreže. Inak povedané, počíta súčet modulo 2^W.

- $\boxed{\alpha - \beta \text{ (priorita 4)}}$ odčíta od čísla α číslo β. Ak je $\alpha < \beta$, spočíta $2^W + \alpha - \beta$, teda rozdiel modulo 2^W.

- $\boxed{\neg\, \alpha \text{ (priorita 9)}}$ spočíta bitovú negáciu čísla α, čo je číslo, ktorého i-ty bit je $\mathbf{0}$ práve vtedy, keď i-ty bit čísla α sa rovná $\mathbf{1}$, a naopak.

- $\boxed{\alpha \wedge \beta \text{ (priorita 8)}, \alpha \vee \beta, \alpha \oplus \beta \text{ (priorita 7)}}$ bitové operácie: *and, or* a *xor.* Vyhodnocujú sa tak, že sa i-ty bit výsledku spočíta z i-teho bitu čísla α a i-teho bitu čísla β podľa nasledujúcich tabuliek:

$$\begin{array}{lll}
\mathbf{0} \wedge \mathbf{0} = \mathbf{0} & \mathbf{0} \vee \mathbf{0} = \mathbf{0} & \mathbf{0} \oplus \mathbf{0} = \mathbf{0} \\
\mathbf{0} \wedge \mathbf{1} = \mathbf{0} & \mathbf{0} \vee \mathbf{1} = \mathbf{1} & \mathbf{0} \oplus \mathbf{1} = \mathbf{1} \\
\mathbf{1} \wedge \mathbf{0} = \mathbf{0} & \mathbf{1} \vee \mathbf{0} = \mathbf{1} & \mathbf{1} \oplus \mathbf{0} = \mathbf{1} \\
\mathbf{1} \wedge \mathbf{1} = \mathbf{1} & \mathbf{1} \vee \mathbf{1} = \mathbf{1} & \mathbf{1} \oplus \mathbf{1} = \mathbf{0}
\end{array}$$

- $\boxed{\alpha << \beta \text{ (priorita 2)}}$ posunie číslo α o β bitov doľava, teda doplní na jeho koniec β núl a odreže prvých β bitov, aby bol výsledok opäť W-bitový.

- $\boxed{\alpha >> \beta \text{ (priorita 2)}}$ posunie číslo α o β bitov doprava, teda doplní na jeho začiatok β núl a odreže posledných β bitov, aby bol výsledok opäť W-bitový.

Ak zátvorky neurčia inak, vyhodnocujú sa operátory s vyššiou prioritou pred operátormi s nižšou prioritou. V rámci rovnakej priority sa vyhodnocuje zľava doprava (s výnimkou operátora $\neg$, ktorý je unárny, a teda sa musí vyhodnocovať zprava doľava).

Príklad: ako fungujú operátory; predpokladáme $W = 4$

$$\begin{aligned}
a + b \wedge c + d &= (a + (b \wedge c)) + d \\
\mathbf{0101} + \mathbf{1110} &= \mathbf{0011} \\
\mathbf{0001} - \mathbf{1111} &= \mathbf{0010} \\
\mathbf{0101} \wedge \mathbf{0011} &= \mathbf{0001} \\
\mathbf{0101} \vee \mathbf{0011} &= \mathbf{0111} \\
\mathbf{0101} \oplus \mathbf{0011} &= \mathbf{0110}
\end{aligned}$$

Ako vyrobiť pomocou $<<$ postupnosť jednotiek:

$$(\mathbf{1} << \mathbf{11}) - \mathbf{1} = \mathbf{1000} - \mathbf{1} = \mathbf{0111}$$

Ako získať z čohokoľvek samé jednotky:

$$a \vee \neg\, a = \mathbf{1111}$$

Výpočet prebieha takto: Najprv sa do registra x nastaví vstup (vždy jedno číslo) a do ostatných registrov nuly. Potom sa vykonajú všetky príkazy *register* := *výraz* v poradí, v akom sú v programe uvedené, pričom sa vždy najskôr vyhodnotí *výraz* na pravej strane a až potom sa jeho výsledok uloží do *registra*, takže vo vnútri výrazu je ešte možné pracovať s pôvodnou hodnotou registra. Po dokončení posledného príkazu sa hodnota v registri y interpretuje ako výsledok výpočtu. Hodnoty v ostatných registroch môžu byť ľubovoľné.

Veľkosť vstupu zadefinujeme ako počet bitov N, ktoré potrebujeme na reprezentáciu vstupu. Naše programy budú často potrebovať registre, do ktorých možno uložiť čísla väčšie ako $2^N - 1$. Počet bitov, ktoré potrebujeme na reprezentáciu týchto čísel budeme volať *veľkosť registra* W. Samozrejme, vždy musí platiť $W \geq N$.

Nie vždy je možné použiť ten istý program pre všetky možné veľkosti vstupu N. V závisloti od N a W je napríklad potrebné zmeniť hodnoty pomocných konštánt v programe, počet opakovaní niektorých operácií v programe a podobne. Preto pri riešení úloh pre každú hodnotu N musíme popísať:

- veľkosť registrov W, ktorú program bude používať, v závislosti od N
- ako pre každú konkrétnu hodnotu N zostaviť program, ktorý funguje pre všetky vstupy veľkosti N

Časovou zložitosťou programu v závislosti od N budeme rozumieť počet inštrukcií, ktoré potrebujeme pre danú veľkosť vstupu. Veľkosť registra W v závislosti od N budeme nazývať *pamäťovou zložitosťou* (keďže počet registrov je obmedzený na 26, veľkosť registra skutočne určuje množstvo pamäte). Tak ako pri časovej a pamäťovej zložitosti bežných programov, pri odhadoch budeme zanedbávať multiplikatívne konštanty (môžeme teda používať O-notáciu). Budeme však vyžadovať, aby **veľkosť registra W bola polynomiálne závislá od veľkosti vstupu N** (t.j. existuje konštanta k, pre ktorú $W \leq N^k$ pre všetky $N \geq 2$).

Pri riešení úloh budeme chcieť, aby časová zložitosť vygenerovaných programov v závislosti na N bola čo najmenšia. Medzi rovnako rýchlymi programami je potom lepší ten s menšou pamäťovou zložitosťou.

Príklad 1: Zostrojte program pre Alik, ktorý dostane na vstupe nenulové číslo a vráti výsledok 1 práve vtedy, ak je toto číslo mocninou dvojky, inak vráti nulu.

Riešenie: Najprv si všimnime, že mocniny dvojky sú práve čísla, ktoré obsahujú práve jeden jednotkový bit. Sledujme chovanie nasledujúceho jednoduchého programu.

Vo všetkých ukážkových programoch budeme v ľavom stĺpci uvádzať jednotlivé príkazy a v pravom stĺpci všeobecný tvar spočítanej hodnoty pre ľubovoľné N. Ak sa nejaká číslica alebo skupina číslic opakuje viackrát, označíme opakovanie exponentom. Teda $\mathbf{0}^8$ je osem núl, $(\mathbf{01})^3$ je skratka za $\mathbf{010101}$. Gréckymi písmenami budeme označovať bližšie neurčené skupiny bitov.

$$\begin{array}{ll} & x = \alpha\mathbf{10}^i \\ a := x - 1 & a = \alpha\mathbf{01}^i \\ b := x \wedge a & b = \alpha\mathbf{00}^i \end{array}$$

Číslo v registri a sa od x vždy líši tým, že najpravejšia $\mathbf{1}$ sa zmení na $\mathbf{0}$ a všetky $\mathbf{0}$ vpravo od nej sa zmenia na $\mathbf{1}$. Preto $b = x \wedge a$ sa musí od x líšiť práve prepísaním najpravejšej $\mathbf{1}$ na $\mathbf{0}$. (To preto, že bity naľavo od tejto $\mathbf{1}$ sú stále rovnaké a $\alpha \wedge \alpha = \alpha$, kým vo zvyšku čísla sa vždy *and*uje $\mathbf{0}$ s $\mathbf{1}$, čo dá nulu.) A keďže mocniny dvojky sú práve čísla, v ktorých dvojkovom zápise je práve jedna $\mathbf{1}$, spočíta náš program v b nulu práve vtedy, ak je x mocnina dvojky (alebo nula, čo sme ale zakázali).

Zostáva teda vyriešiť, ako z nuly spraviť požadovanú jednotku a z nenuly nulu. K tomu si zavedieme operáciu $r := \mathit{if}(s, t, u)$, ktorá bude realizovať podmienku: ak $s \neq 0$, priradí $r := t$, inak $r := u$. Spravíme to jednoduchým trikom: rozšírime si registre o jeden pomocný bit vľavo, nastavíme v r tento bit na jednotku a sledujeme, či sa zmenšením vzniknutého čísla o jednotku tento bit zmení na nulu alebo nie:

$$\begin{array}{ll} v := s \vee \mathbf{10}^N & v = \mathbf{1}r \\ v := v - 1 & v = \mathbf{1}r' \text{ (ak } r \neq 0\text{), inak } \mathbf{01}^N \\ v := v \wedge \mathbf{10}^N & v = \mathbf{10}^N \text{ alebo } \mathbf{00}^N \\ v := v >> N & v = \mathbf{0}^N\mathbf{1} \text{ alebo } \mathbf{0}^N\mathbf{0} \\ v := v - 1 & v = \mathbf{0}^{N+1} \text{ alebo } \mathbf{1}^{N+1} \\ r := (u \wedge v) \vee (t \wedge \neg v) & s = t \text{ alebo } u \end{array}$$

Stačí teda na koniec programu pridať

$$y := \mathit{if}(b, 0, 1) \qquad y = \mathbf{0} \text{ alebo } \mathbf{1}$$

a máme program, ktorý rozpoznáva mocniny dvojky v konštantnom čase a používa na to čísla s $N + 1 = O(N)$ bitmi.

Ešte si ukážme, ako bude prebiehať výpočet pre dva konkrétne 8-bitové vstupy (teda $N = 8$ a $W = 9$):

	$x = \mathbf{001011000}$	$x = \mathbf{000100000}$
$a := x - 1$	$a = \mathbf{001010111}$	$a = \mathbf{000011111}$
$b := x \wedge a$	$b = \mathbf{001010000}$	$b = \mathbf{000000000}$
$v := b \vee \mathbf{100000000}$	$v = \mathbf{101010000}$	$v = \mathbf{100000000}$
$v := v - 1$	$v = \mathbf{101001111}$	$v = \mathbf{011111111}$
$v := v \wedge \mathbf{100000000}$	$v = \mathbf{100000000}$	$v = \mathbf{000000000}$
$v := v >> 8$	$v = \mathbf{000000001}$	$v = \mathbf{000000000}$
$v := v - 1$	$v = \mathbf{000000000}$	$v = \mathbf{111111111}$
$y := (\mathbf{000000001} \wedge v)$ $\vee(\mathbf{000000000} \wedge \neg v)$	$y = \mathbf{000000000}$	$y = \mathbf{000000001}$

Príklad 2: Zostrojte program pre Alik, ktorý spočíta *binárnu paritu* vstupného čísla, teda vráti 0 nebo 1 podľa toho, či má toto číslo párny alebo nepárny počet jednotkových bitov.

Riešenie: Bez ujmy na všeobecnosti môžeme predpokladať, že N je mocnina dvoch. Binárna parita $P(x)$ čísla $x = x_{N-1} \dots x_1 x_0$ sa podľa definície rovná $x_0 \oplus x_1 \oplus \dots \oplus x_{N-1}$. Keďže operácia $\oplus$ je asociatívna ($\alpha \oplus (\beta \oplus \gamma) = (\alpha \oplus \beta) \oplus \gamma$) a komutatívna ($\alpha \oplus \beta = \beta \oplus \alpha$), môžeme tento vzťah preusporiadať na

$$P(x) = (x_0 \oplus x_{N/2}) \oplus (x_1 \oplus x_{N/2+1}) \oplus \dots \oplus (x_{N/2-1} \oplus x_{N-1}),$$

čo je ale parita čísla, ktoré vznikne vyxorovaním ľavej a pravej polovice čísla x. Takže výpočet parity N-bitového čísla môžeme konštantným počtom príkazov previesť na výpočet parity $N/2$-bitového čísla, ten zase na výpočet parity $N/4$-bitového čísla atď., až po $\log_2 N$ krokoch na paritu 1-bitového čísla, ktorá sa ovšem rovná číslu samotnému.

Paritu teda spočítame na logaritmický počet príkazov pracujúcich s N-bitovými číslami takto:

$p := x >> N/2$	$p =$ horných $N/2$ bitov x
$q := x \wedge \mathbf{1}^{N/2}$	$q =$ dolných $N/2$ bitov x
$x := p \oplus q$	$x = N/2$-bitové číslo s paritou ako pôvodné x
$x := (x >> N/4) \oplus (x \wedge \mathbf{1}^{N/4})$	$x = N/4$-bitové číslo s rovnakou paritou (všimnite si skrátený zápis výrazu)
...	...
$x := (x >> 1) \oplus (x \wedge \mathbf{1})$	$x =$ 1-bitové číslo
$y := x$	$y = x$ (skopírovať výsledok)

Časová zložitosť tohto programu je $O(\log N)$ krokov. Pamäťová zložitosť je $O(N)$ bitov.

Náš programovací jazyk samozrejme žiadne celé časti čísel a podobné operácie nemá, ale to vôbec nevadí, pretože ich vždy používame len na podvýrazy závisiace iba na N, takže ich v programe môžeme pre každé N uviesť ako konštanty. Napríklad pre $N = 8$ bude výpočet prebiehať takto:

$$
\begin{array}{ll}
 & x = \mathbf{00110110} \\
p := x >> 4 & p = \cdots\cdot\mathbf{0011} \\
q := x \wedge \mathbf{1111} & q = \cdots\cdot\mathbf{0110} \\
x := p \oplus q & x = \cdots\cdot\mathbf{0101} \\
x := (x >> 2) \oplus (x \wedge \mathbf{11}) & x = \cdots\cdots\mathbf{00} \\
x := (x >> 1) \oplus (x \wedge \mathbf{1}) & x = \cdots\cdots\cdot\mathbf{0} \\
y := x & y = \mathbf{00000000}
\end{array}
$$

Súťažné úlohy

a) Zostrojte program pre Alik, ktorého výsledkom bude počet jednotkových bitov v dvojkovom zápise čísla na vstupe.

Príklad: Pre vstup $x = \mathbf{101}$ program vráti $y = \mathbf{10}$.

b) Zostrojte program pre Alik, ktorý pre zadané číslo x vypočíta najbližšie väčšie číslo, ktoré má v dvojkovom zápise rovnaký počet jednotiek ako x. Ak také číslo neexistuje, výsledok môže byť ľubovoľný.

Príklad: Pre vstup $x = \mathbf{1010}$ program vráti $y = \mathbf{1100}$.

SLOVENSKÁ KOMISIA MATEMATICKEJ OLYMPIÁDY

54. ROČNÍK MATEMATICKEJ OLYMPIÁDY

Zadania 1. kola kategórie P

Vydala IUVENTA pre vnútornú potrebu Ministerstva školstva SR
Zodpovedný redaktor: M. Forišek
Sadzba programom LaTeX