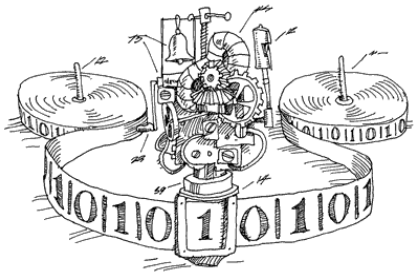
Úloha č. 0b
Lexikální analyzátor #2

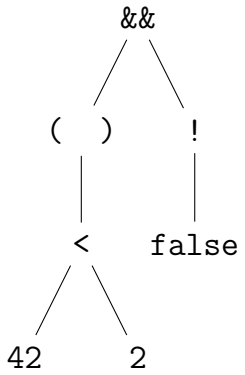


Odpověz Sfinze! 2.5/10 b

Tato úloha je vyhodnocována automaticky. Je potřeba, aby výstup programu ***přesně*** *korespondoval se specifikací výstupu níže. Jak odevzdávat tento typ úloh se můžeš dočíst na webových stránkách FIKSu pod záložkou „Jak řešit FIKS“.*
Tato úloha je součástí seriálu, tzn. že bude otevřena až do konce ročníku, ale každé další kolo se jí zmenší maximální možné ohodnocení na polovinu, proto bys ji měl vyřešit co nejdříve.Pokud jsi tuto úlohu řešil už v minulých kolech, tak ji můžeš s klidem přeskočit. Tato úloha byla zveřejněna v 2. kole, tzn. že její ***maximální zisk je hodnocen pouze 25 %****.*

Praporčík Paxton se jistě brzy dočká povýšení za to, jak nedávno naprogramoval řádkový vstup pro simulaci přistávacích manévrů. Operace Mars má velký úspěch a Paxton na tom má velký podíl. Sám Paxton z toho ale tak veselý není. Z povrchu Marsu neustále přicházejí dotazy na hlavní počítač, který se pod tou tíhou zhroutil a praporčík ho musel restartovat. Při tom se ale ztratila důležitá část vstupního modulu. Lodní počítač přestal načítat a vyhodnocovat logické výrazy. Praporčík ho bude muset napsat celý znovu.

Vstupní řádka obsahuje opět *lexémy*. Tentokrát je však syntaktický význam lexémů složitější. Již se nedá spoléhat na jejich pořadí. Na začátku očekáváme jeden celý výraz. Pokud je na řádce například jen číslo, máme hotovo. Může zde být ale unární operátor (negace), za kterým nutně musí následovat jeden další podvýraz. V případě binárního operátoru budou podvýrazy dva – levý a pravý. Speciálním případem jsou potom závorkové struktury. Otevírací závorka nám vlastně nastartuje celý proces znovu, za závorkou očekáváme právě jeden výraz, za kterým musí být závorka uzavírací. Závorky se tak vlastně chovají jako takový „dvojdílný“ unární operátor. Konec řádky se rozpozná jednoduše, přečetli jsme právě tolik podvýrazů, kolik bylo vyžadováno. Celý výraz si tedy můžeme představit jako strom (viz obrázek 0.1).



Obrázek 0.1 Příklad *syntaktického stromu* pro výraz `(42 < 2) && !false`

No a jak se takový strom vyhodnocuje? K vyhodnocení libovolného uzlu potřebuji znát hodnotu všech jeho potomků. Pokud je uzel listem, jeho hodnotu znám. Stačí tedy začít od listů a postupovat směrem vzhůru ke kořeni.

Vstup

Každý vstup obsahuje N řádků, každý z nich reprezentuje jedno zadání. Řádek se zadáním obsahuje jeden logický výraz.

Logickým výrazem rozumíme následující:

- term,
- term $\mathcal{OP}$ term, kde $\mathcal{OP}$ je jeden z operátorů `&&`, `||`, `==`, `!=`,
- numerický výraz.

Pod pojmem term rozumíme:

- číslo a (při vyhodnocování má $a = 0$ logickou hodnotu `false`, jinak `true`),
- logickou hodnotu `true` nebo `false`,
- negaci termu operátorem `!`,
- řádně uzávorkovaný logický výraz.

Numerický výraz je ve formátu a $\mathcal{OP}$ b, kde a, b jsou celá čísla a $\mathcal{OP}$ je jeden z následujících operátorů: `<`, `<=`, `>=`, `>`. Maximální rozsah a, b a velikost N jsou $100 < N \leq 1\,000$ a $0 \leq a, b \leq 2^{30}$. Vstupní lexémy nemusí být odděleny mezerou.

Výstup

Výstup obsahuje N řádků. Každý řádek odpovídá jednomu zadání. Obsahem každého řádku je řetězec `true` nebo `false` charakterizující logickou hodnotu, která odpovídá vyhodnocení výrazu.

Ukázkové vstupy

Vstup	Výstup
`true`	`true`
`0`	`false`
`27 < 2`	`false`
`42 || (false)`	`true`
`!false && (23 >= 3)`	`true`
`(true != true) || ((30 > 2) && !(29 <= 1))`	`true`
`true != !(!!!true)`	`false`