ZADÁNÍ SEMESTRÁLNÍ PRÁCE
VYHODNOCOVÁNÍ LOGICKÝCH FORMULÍ

Zadání

Naprogramujte v ANSI C přenositelnou[1] **konzolovou aplikaci**, která jako vstup načte z parametru na příkazové řádce logickou (predikátovou) formuli a na výstup vypíše tabulku pravdivostních hodnot pro všechny kombinace hodnot proměnných a určí, zda se jedná o *tautologii*, případně *kontradikci*, pokud některá z těchto eventualit nastane.

Program se bude spouštět příkazem `bools.exe` <form>. Symbol <form> zastupuje zápis logické formule (způsob zápisu bude upřesněn dále). Takže Váš program může být během testování spuštěn například takto:

```
bools.exe (a&b)|(~b->c)
```

Výstupem programu bude tabulka pravdivostních hodnot v přehledné podobě uvedené níže. Pokud nebude uveden právě jeden argument, vypište chybové hlášení a stručný návod k použití programu (v angličtině).

Hotovou práci odevzdejte v jediném archivu ZIP, pojmenovaném Vaším osobním číslem, tj. např. `A03487.zip`. Archiv nechť obsahuje všechny zdrojové soubory potřebné k přeložení programu, **makefile** pro Windows i Linux a dokumentaci ve formátu PDF nebo PostScript. Bude-li některá z částí chybět, kontrolní skript Vaši práci odmítne.

Specifikace výstupu programu

Výstup analyzátoru bude směrován jak na konzoli, tak do souboru. **Pokud bude analyzovaná logická formule obsahovat více než 5 proměnných, provede se výstup pouze do souboru**. Soubor s výstupem[2] pojmenujte podle data a času spuštění analyzátoru podle vzoru `YYYY-MM-DD-HHMMSS.bool`. Následující výpis ukazuje, jak bude vypadat výstup na konzoli:

```
X:\>bools.exe (~a|~b)==~(a&b)
PARSER-RSL=OK; VARS=2; D$=4;

 A | B | $
---+---+---
 0 | 0 | 1
 0 | 1 | 1
 1 | 0 | 1
 1 | 1 | 1

TAUTOLOGY
```

Na první řádce výstupu analyzátoru budou statistické údaje:
(i) výsledek analýzy výrazu, a to `PARSER-RSL=OK` v případě, že je vstup syntakticky v pořádku a byl bez chyby analyzován, nebo `PARSER-RSL=ERR;` v případě chyby. Popis chyby může být v závorce, např. `PARSER-RSL=ERR(missing operator);`

[1] Je třeba, aby bylo možné Váš program přeložit a spustit na PC s operačním prostředím Win32 (tj. operační systémy Microsoft Windows 95/98/NT/ME/2000/XP/Vista) a s běžnými distribucemi Linuxu (např. Ubuntu, Fedora Core, Gentoo, Debian, ...). Server, na který budete Vaši práci nahrávat a který ji otestuje, je Sun Blade 100 s 64-bitovým procesorem SPARC a je na něm provozován Gentoo Linux s překladačem GCC 3.3.5.

[2] Obsah souboru je identický s výstupem na konzoli v případě nejvýše 5 proměnných.

(ii) počet identifikovaných logických proměnných, např. `VARS=2;`
(iii) mohutnost formule (tedy počet řádek pravdivostní tabulky), např. `D$=4;` – tato hodnota je zřejmě 2^{VARS}.

Po řádce se statistickými údaji následuje jedna prázdná řádka a pak pravdivostní tabulka. Tabulku formátujte podle výše uvedeného vzoru. Identifikované proměnné převeďte na velká písmena – pokud budou jejich identifikátory obsahovat více znaků, pak převeďte na velká písmena všechny (analyzátor tedy **nebude** case-sensitive). Logické hodnoty zarovnávejte pod písmeno proměnné, resp. pod nejpravější písmeno víceznakového identifikátoru. Výsledek celé formule bude v záhlaví tabulky označen znakem `$` (dolar).

Pod tabulkou bude opět jedna prázdná řádka a pak velkými písmeny výraz označující typ analyzované formule (anglicky), a to buď `TAUTOLOGY`, `CONTRADICTION` nebo `NONSPECIFIC`.

Způsob zápisu formulí

Tabulka uvádí přehled všech operátorů, které je možné v zápisu logické formule použít:

Logická operace	Zápis operátoru	Příklad
negace (not)	`~`	`~a`
konjunkce (and)	`&`	`a & b`
disjunkce (or)	`|`	`a | b`
implikace	`->`	`a -> b`
ekvivalence	`==`	`a == b`
nonekvivalence (xor)	`^`	`a ^ b`

Jiné operátory nejsou povolené. Priorita operátorů je dána vlastnostmi booleovské algebry, případně upravena závorkami `(` a `)`. Např.: `((a -> ~b) & (c == a)) | (~a & b)`.

Jednotlivé logické proměnné ve výrazu se dopředu nedeklarují. Za deklaraci se považuje první výskyt takové proměnné ve zpracovávaném výrazu. Při řešení úlohy počet proměnných neomezujte – navrhněte takový algoritmus, který jich dokáže zpracovat libovolné množství (nebo jako omezující počet volte dostatečně vysoké číslo, např. 256 proměnných[3]).

Uvědomte si, že logická formule se bude analyzátoru předávat jako argument na příkazové řádce, tj. zápis nesmí obsahovat mezery, jinak by byl příkazovým procesorem (ve Windows program `cmd`) vyhodnocen jako několik argumentů. Jedním ze způsobů, jak přinutit příkazový procesor brát řetězec obsahující mezery jako jeden argument, je uzavřít jej do uvozovek `"..."`. Umožněte oba dva způsoby zápisu, tedy „hustý“ zápis bez mezer i zápis s mezerami uzavřený do uvozovek. Obě tato volání tudíž skončí se stejným výsledkem:

```
X:\>bools.exe ~(a->b)&~(a|~c)
X:\>bools.exe "~(a -> b) & ~(a | ~c)"
```

Užitečné techniky

Uvedené techniky je možné (ale nikoliv nezbytně nutné) využít při řešení úlohy. Protože se jedná o postupy víceméně standardní, lze k nim nalézt velké množství dokumentace:

1. Syntaktická analýza rekurzivním sestupem
2. Binární strom

Řešení úlohy je zcela ve vaší kompetenci – zvolte takové algoritmy a techniky, které podle vás nejlépe povedou k cíli...

[3] Výsledná tabulka takové funkce by pak měla zřejmě 2^{256} řádek, což je opravdu hodně...