FLP 2022/2023 – logický projekt: Prolog

Toto je zadání logického projektu do předmětu Funkcionální a logické programování 2022/2023. Za projekt zodpovídá **Ing. Radek Hranický, Ph.D.** Konzultace jsou možné kdykoliv po předchozí domluvě; případně elektronickou poštou (`hranicky@fit.vut.cz`) či prostřednictvím diskusního fóra předmětu, které najdete v IS VUT v sekci e-Learning.

Za vypracovaný projekt lze získat až 8 bodů. Minimální počet bodů pro získání zápočtu jsou 3. **Hodnocena bude míra splnění zadání, kvalita řešení, čistota a kvalita kódu a vhodné užití komentářů.** Za inovativní přístup či obzvláště kvalitní řešení lze získat prémiové body navíc.

Obecné pokyny pro všechna zadání jsou sepsány ve zvláštním dokumentu, nezapomeňte na jejich dodržování. **Projekt je nutné vypracovat samostatně –** duplikáty jsou kontrolovány i strojově a pokus o opsání nebo přílišnou "inspiraci" cizím kódem je velmi dobře poznat! Případné nejasnosti konzultujte včas, na námitky po odevzdání nebude brán zřetel. U všech zadání je kladen důraz především na správné chování v případě správného vstupu. Cílem projektu je vyzkoušet si principy logického programování, nikoliv dokonalé ošetření vstupu. Obecně také platí, že pokud existuje cesta k vyřešení problému, Váš program ji musí nalézt. Pokud existuje více cest, může Váš program zvolit libovolnou z nich.

Úvod všech zdrojových textů musí obsahovat *název projektu*, *login* a *jméno autora*. Součástí řešení je Makefile (projekty budou překládány pomocí `make`), cílový program pojmenujte **`flp22-log`**. Makefile ani přeložený program **neumisťujte do podadresářů**.[1] Projekt vypracujte v prostředí SWI-Prolog, za referenční stroj je považován server `merlin`. **Pokud nebude projekt funkční na serveru `merlin`, nebudou uděleny body. Před odevzdáním vždy vyzkoušejte spuštění projektu na serveru `merlin`!**

Spolu s projektem odevzdejte i několik vstupních souborů, na nichž je možné váš projekt otestovat, do dokumentace uveďte pro každý takový soubor přibližnou dobu výpočtu. Prosím neodevzdávejte zbytečné soubory.

Následuje popis jednotlivých variant zadání.

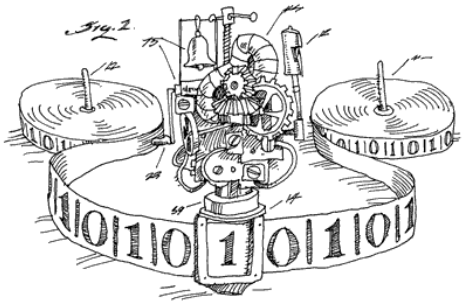
[1] Tzn. např. sekvencí příkazů

```
unzip flp-log-xlogin00.zip
make
./flp22-log < /cesta_k_testum/vstup1.txt > /cesta_k_vysledkum/vystup1.txt
```

dojde k rozbalení odevzdaného archivu, zkompilování projektu a spuštění na testovacích datech.

(verze zadání 2023-02-14)

Turingův stroj



Vaším úkolem je vytvořit simulátor nedeterministického Turingova stroje. Na vstupu váš program obdrží pravidla pro Turingův stroj a vstupní obsah pásky. Výstupem bude posloupnost konfigurací stroje.

Vnitřní stavy stroje jsou označeny velkými písmeny, vstupní/páskovou abecedu tvoří malá písmena, prázdný symbol je mezera, počáteční stav je „S“, koncový stav je „F“. Výpočet stroje začíná na začátku pásky a končí přechodem do koncového stavu nebo abnormálním zastavením. Pravidla jsou zadána ve tvaru <stav> <symbol na pásce> <nový stav> <nový symbol na pásce nebo „L“, „R“>. Jednotlivé části jsou odděleny mezerou, každé pravidlo bude na samostatném řádku. Symboly L/R značí posun hlavy doleva/doprava. Na posledním řádku vstupu je uveden vstupní obsah pásky (nekonečná posloupnost prázdných symbolů na konci pásky není uvedena).

Jednotlivé konfigurace stroje vypisujte na samostatné řádky. Konfiguraci uvádějte v pořadí <obsah pásky před hlavou><stav><symbol pod hlavou><zbytek pásky> (bez oddělovačů).

Zaměřte se na korektní vstupy a situace, kdy existuje sekvence přechodů do koncového stavu. V takovém případě Váš program musí toto řešení najít. Implementační detaily v případě abnormálního zastavení nebo cyklení jsou ponechány na autorovi.

Příklad vstupu a výstupu:

```
S a B a
B a B b
B b B R
B c B a
B c F c
B c B a
aaacaa
```

```
Saaacaa
Baaacaa
Bbaacaa
bBaacaa
bBbacaa
bbBacaa
bbBbcaa
bbbBcaa
bbbFcaa
```

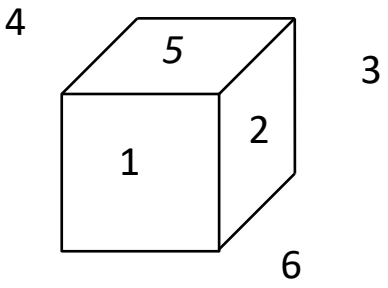
Rubikova kostka

Rubikova kostka je hlavolam ve tvaru krychle, každá strana je vybarvena odlišnou barvou a je rozdělena na 3x3 dílů, jednotlivé vrstvy lze vzájemně pootočit o násobek 90 °. Cílem je kostku, která byla zamíchána náhodným otáčením jednotlivých vrstev, uvést do složeného stavu (každá strana jednou barvou).

Vstupem vašeho programu bude popis zamíchané kostky, výstupem návod, jak ji složit. Návod bude obsahovat popis kostky po každém kroku řešení, jednotlivé popisy oddělené prázdným řádkem. Nejprve uveďte popis zadané zamíchané kostky, pak postupně jednotlivé kroky řešení.

Jednotlivé barvy v každém dílku jsou popsány číslicemi 1—6, každá strana kostky je tedy popsána devíti číslicemi. Číslice z každé strany jsou zapisovány po řádcích a jednotlivé strany jsou uvedeny v pořadí podle následujícího schématu. Číslice ve skupinkách nejsou nijak odděleny, nejprve je uvedena horní strana kostky, pak přední, pravá, zadní a levá odděleny mezerou a pak spodní strana (viz následující ukázka – nákres kostky, její plášť a zápis v požadovaném formátu).

Ukázka složené kostky:



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

```
555
555
555
111 222 333 444
111 222 333 444
111 222 333 444
666
666
666
```

Příklad vstupu:

```
553
553
554
225 322 644 111
115 322 633 444
115 322 633 444
662
661
661
```

Příklad výstupu:

```
553
553
554
225 322 644 111
115 322 633 444
115 322 633 444
662
661
661

555
555
555
222 333 444 111
111 222 333 444
111 222 333 444
666
666
666

555
555
555
111 222 333 444
111 222 333 444
111 222 333 444
666
666
666
```

Babylonská věž

Babylonská věž je hlavolam ve tvaru válce složeného z otočných prstenců, kolmo na prstence jsou umístěny výřezy, ve kterých jsou umístěny různobarevné kuličky. Kuličky v jednom sloupci (výřezu) mají všechny stejný odstín barvy, jsou seřazeny od nejsvětlejší po nejtmavší. Otáčením prstenců je možné přesouvat kuličky mezi sloupci, v jednom prstenci je možné přesunout jednu kuličku v každém výřezu. V jednom výřezu je o kuličku méně, vznikl tak volný prostor, pomocí něhož je možné přesouvat kuličky mezi prstenci.

Vstupem vašeho programu bude popis zamíchané věže, výstupem návod, jak ji složit. Návod bude obsahovat popis věže po každém kroku řešení, jednotlivé popisy oddělené prázdným řádkem. Nejprve uveďte popis zadané zamíchané věže, pak postupně jednotlivé kroky řešení.

Každá kulička je popsána barvou – označena popořadě písmeny A—Z (bude se vyskytovat nejvýše 26 barev) – a úrovní odstínu (číslovány od 1). Aktuální rozměry věže musíte odvodit z jejího popisu. Volný prostor je označen dvěma hvězdičkami **, nachází se za poslední kuličkou sloupce A. Každý prstenec je vypsán na samostatný řádek, kuličky jsou odděleny mezerami. Příklad věže se třemi barvami a čtyřmi prstenci, chybí kulička A4 a kuličky A2, A3 jsou posunuty o pozici níže:

```
A1 B1 C1
** B2 C2
A2 B3 C3
A3 B4 C4
```

Ukázka vstupu:

```
D1 E1 F1 A1 B1 C1
F2 E3 B2 C2 D2 E2
D3 E4 F3 A2 B3 C3
A3 B4 C4 D4 ** F4
```

Příklad požadovaného výstupu:

```
D1 E1 F1 A1 B1 C1
F2 E3 B2 C2 D2 E2
D3 E4 F3 A2 B3 C3
A3 B4 C4 D4 ** F4

A1 B1 C1 D1 E1 F1
C2 D2 E2 F2 E3 B2
A2 B3 C3 D3 E4 F3
A3 B4 C4 D4 ** F4
```

```
A1 B1 C1 D1 E1 F1
C2 D2 E2 F2 ** B2
A2 B3 C3 D3 E3 F3
A3 B4 C4 D4 E4 F4

A1 B1 C1 D1 E1 F1
** B2 C2 D2 E2 F2
A2 B3 C3 D3 E3 F3
A3 B4 C4 D4 E4 F4

A1 B1 C1 D1 E1 F1
A2 B2 C2 D2 E2 F2
A3 B3 C3 D3 E3 F3
** B4 C4 D4 E4 F4
```

Kostra grafu

Vašim úkolem je napsat program schopný nalézt všechny kostry zadaného neorientovaného grafu. Kostra je takový podgraf, který je tvořen všemi vrcholy původního grafu a zároveň je to strom (je souvislý a neobsahuje žádnou kružnici). Na vstupu budete mít zadány neorientované hrany grafu ve formě dvojic oddělených mezerou. Tato písmena značí vrcholy grafu. Pokud by vstupní graf nebyl souvislý, tak se na výstup nevytiskne nic.

Vrcholy jsou vždy označeny pomocí velkých písmen anglické abecedy (A až Z). Jednotlivé řádky vstupního souboru mají tvar „<V1> <V2>“ (bez uvozovek). Jakýkoliv jiný řádek je ignorován. Jako výstup programu se očekává seznam koster, kde na jednom řádku je jedna kostra. Každá kostra má výstup ve tvaru „<V1>-<V2> <V3>-<V4> … <Vn-1>-<Vn>”. Každá dvojice vrcholů reprezentuje hranu dané kostry. Pořadí vrcholů v hraně, jednotlivých hran, nebo pořadí koster ve výstupu je libovolné, každá kostra však musí být unikátní.

Ukázka vstupu a výstupu:

A B A C A D B C C D
A-B A-C A-D A-B A-C C-D A-B A-D B-C A-B A-D C-D A-B B-C C-D A-C A-D B-C A-C B-C C-D A-D B-C C-D

Hamiltonovská kružnice

Vašim úkolem je napsat program schopný nalézt všechny Hamiltonovské kružnice zadaného neorientovaného grafu. Hamiltonovská kružnice prochází každým vrcholem grafu právě jednou přičemž začíná a končí v tom samém vrcholu. Na vstupu budete mít zadány neorientované hrany grafu ve formě dvojic oddělených mezerou. Tato písmena značí vrcholy grafu. Pokud by vstupní graf nebyl souvislý, tak se na výstup nevytiskne nic.

Vrcholy jsou vždy označeny pomocí velkých písmen anglické abecedy (A až Z). Jednotlivé řádky vstupního souboru mají tvar „<V1> <V2>“ (bez uvozovek). Jakýkoliv jiný řádek je ignorován. Jako výstup programu se očekává seznam Hamiltonovských kružnic, kde na jednom řádku je jedna kružnice. Každá kružnice má výstup ve tvaru „<V1>-<V2> <V3>-<V4> … <Vn-1>-<Vn>”. Každá dvojice vrcholů reprezentuje hranu dané kružnice. Pořadí vrcholů v hraně, jednotlivých hran, nebo pořadí kružnic ve výstupu je libovolné, každá kružnice však musí být unikátní.

Ukázka vstupu a výstupu:

A B A C A D B C B D C D
A-B A-C B-D C-D A-B A-D B-C C-D A-C A-D B-C B-D

Vlastní zadání

Vymyslete si a implementujte vlastní zadání. **Vaše zadání musí být předem schváleno opravujícím.** Rozhodnutí o schválení či zamítnutí zadání záleží pouze na opravujícím. Zadání by mělo být odlišné od zadání řešených v minulých letech i od úkolů logického programování řešených v jiných předmětech. Zadání musí být alespoň tak obtížné, jako jsou ta zde uvedená.