71. ročník Matematické olympiády – 2021/2022

Úlohy školního kola kategorie P

Řešení úloh odevzdávejte pomocí webového rozhraní, které je dostupné na stránkách olympiády *https://mo.mff.cuni.cz/*. Tam také najdete podrobnější instrukce k odevzdávání a další informace o kategorii P.

Úlohy P-I-1 a P-I-2 jsou praktické. Vaším úkolem v nich je vytvořit a odladit efektivní program v jazyce Pascal, C nebo C++. Řešení těchto dvou úloh odevzdávejte přes webové rozhraní ve formě zdrojového kódu. Odevzdaná řešení budou automaticky vyhodnocena pomocí připravených vstupních dat a výsledky vyhodnocení se dozvíte krátce po odevzdání. Pokud váš program nezíská plný počet 10 bodů, můžete své řešení opravit a znovu odevzdat.

Úlohy P-I-3 a P-I-4 jsou teoretické, za každou z nich lze získat až 10 bodů. Řešení musí obsahovat popis algoritmu, zdůvodnění jeho správnosti a v úloze P-I-3 též odhad časové a paměťové složitosti. Nemusíte psát program, algoritmus stačí zapsat ve vhodném pseudokódu nebo dokonce jenom slovně, ale v tom případě dostatečně podrobně a srozumitelně. Hodnotí se nejen správnost, ale také efektivita zvoleného postupu řešení. Řešení obou teoretických úloh odevzdávejte ve formě souboru typu PDF přes výše uvedené webové rozhraní.

Řešení všech úloh můžete odevzdávat do 15. listopadu 2021. Opravená řešení a seznam postupujících do krajského kola najdete na webových stránkách olympiády.

Experimentální jazyky: V letošním ročníku MO-P je také možné odevzdávat praktické úlohy i v jazycích Python 3 a Java 11. U nich nicméně nezaručujeme, že bude možné získat plný počet bodů – je možné, že program nestihne doběhnout do časového limitu, byť používá algoritmus s optimální časovou složitostí.

P-I-1 (Ne)tančící starosta

Na kocourkovský taneční ples právě dorazil fotograf, aby pořídil každoroční slavnostní snímek všech tanečníků. Momentálně všech n tanečníků stojí v řadě a zleva doprava si je očíslujeme čísly 1 až n. Výšku tanečníka na pozici i označíme h_i. Tanečník číslo s je starosta.

Na snímku jsou tradičně tanečníci seřazení podle výšky v neklesajícím pořadí. A jelikož jde o ples, tanečníci mohou měnit pořadí pouze jedním způsobem, pomocí tanečních figur. Každá figura vypadá tak, že se nějaký souvislý úsek tanečníků zatočí a skončí v přesně opačném pořadí, než začínal.

Pokud například začneme s tanečníky s výškami

$$(180, 185, 183, 182, 181, 185, 190),$$

lze je uspořádat pomocí jedné taneční figury, do nichž se zapojí tanečníci na pozicích 2 až 5.

Je veřejným tajemstvím, že starosta vůbec neumí tancovat. Abychom ho neztrapnili, je potřeba, aby se vůbec nehýbal – po každé figuře musí starosta zůstat na svém místě.

Soutěžní úloha

Pro daný popis situace zjistěte, zda vůbec lze tanečníky uspořádat, aby starosta vždy zůstal na svém místě. Pokud ano, najděte nějaký způsob, jak to udělat pomocí „rozumně malého“ počtu tanečních figur.

Přesněji řečeno, v každém vstupu navíc dostanete nějakou hodnotu ℓ. Pokud $\ell = 0$, stačí, když zjistíte, zda řešení existuje. Pokud je $\ell > 0$, pro řešitelné vstupy také sestrojte nějaké řešení používající nejvýše ℓ figur.

Vždy bude platit, že v řešitelných vstupech, kde je $\ell > 0$, existuje nějaké řešení s nejvýše ℓ figurami. Libovolné řešení, které tento limit splňuje, bude přijaté, nemusíte minimalizovat počet figur.

Formát vstupu

Na prvním řádku dostanete tři mezerou oddělená celá čísla n, s a ℓ. Na druhém řádku dostanete mezerou oddělená čísla $h_1, \ldots, h_n$.

Formát výstupu

Na prvním řádku vypište řetězec `ANO`, pokud tanečníky lze uspořádat, resp. `NE`, pokud to nelze.

Je-li $\ell > 0$, následuje ještě druhá část výstupu. Ta začíná řádkem obsahující jedno celé číslo $f \leq \ell$: počet tanečních figur, které chcete provést. Na každém z následujících f řádků popište postupně jednotlivé figury (od první po f-tou). Pro každou figuru vypište řádek obsahující dvě mezerou oddělená celá čísla z_j, k_j, kde $1 \leq z_j \leq k_j \leq n$ jsou pozice začátku a konce úseku, jehož pořadí má vaše taneční figura obrátit (včetně z_j a k_j, tj. tanečník začínající na pozici z_j skončí na pozici k_j a naopak).

Příklady

Vstup:	*Výstup:*
`5 3 100` `160 165 175 170 180`	`NE`

Starosta stojí uprostřed. Jelikož se nesmí hýbat, řešení zřejmě neexistuje.

Vstup:	*Výstup:*

`5 3 100`
`185 180 175 170 165`
	`ANO`
	`1`
	`1 5`

Starosta opět stojí uprostřed. Tentokrát však můžeme otočit celou řadu tanečníků. Všimněte si, že tato figura obsahuje i starostu, ten však během ní celou dobu zůstane na svém místě, takže je všechno v pořádku.

Vstup:	*Výstup:*
`5 3 0` `175 170 175 180 175`	`ANO`

Tento vstup má $\ell = 0$, je tedy jen třeba rozhodnout, zda má, či nemá řešení.

Vstup:	*Výstup:*

`5 3 100`
`175 170 175 180 175`
	`ANO`
	`4`
	`4 5`
	`1 2`
	`4 5`
	`4 5`

Bodování

Ve všech vstupech platí $1 \leq n \leq 1000$, $1 \leq s \leq n$ a $1 \leq h_i \leq 10^9$ pro všechna $1 \leq i \leq n$.

Je pět sad vstupů, za vyřešení každé z nich dostanete dva body. V rámci každé sady mají všechny vstupy stejnou hodnotu ℓ, zároveň se v následující tabulce pro každou sadu dozvíte také horní odhad na n pro všechny vstupy v dané sadě:

body	n	ℓ
2	1000	0
2	20	1000
2	100	2000
2	1000	3000
2	1000	1500

P-I-2 To je ale rozhled!

Prominence neboli *význačnost* hory popisuje, nakolik vyčnívá z okolní krajiny. Má více navzájem ekvivalentních definic, jedna z nich vypadá takto:

- Nejvyšší hora na světě má prominenci rovnou své nadmořské výšce.
- Pro každou další horu platí, že pokud v je její nadmořská výška, tak její prominence je nejmenší takové p, že pokud začneme túru na vrcholu naší hory a chceme se dostat na nějaké výše položené místo, musíme někdy po cestě klesnout do nadmořské výšky $v - p$.

Například Ještěd (1012 m n. m.) má prominenci 517 metrů. Pokud začneme výlet na Ještědu a nikdy neklesneme na $1012 - 517 = 495$ m n. m., zůstaneme pouze na Ještědsko-kozákovském hřbetu, jehož je Ještěd nejvyšší horou. Pokud však přejdeme skrz Jeřmanice (495 m n. m.), umíme se potom okolo Jablonce n. Nisou dostat až do Jizerských hor či Krkonoš, a tam vystoupat do vyšší nadmořské výšky, než má Ještěd (např. cestou na Sněžku).

Soutěžní úloha

Z moře trčí jeden horský hřeben. Předpokládáme, že to je jediná pevnina na celém světě. Hřeben můžeme popsat jako lomenou čáru určenou body se souřadnicemi $(0, 0), (1, h_1), (2, h_2), \ldots, (n, h_n), (n+1, 0)$. První souřadnice je vodorovná, druhá je svislá.

Hory jsou lokální maxima hřebenu, tj. ty jeho body a vodorovné úseky, z nichž hřeben na obě strany pokračuje dolů. Pro každou horu spočítejte její prominenci.

Formát vstupu

Na prvním řádku je celé číslo n. Na druhém řádku jsou mezerou oddělená celá čísla $h_1, \ldots, h_n$ popisující hřeben.

Formát výstupu

Pro každou horu (v pořadí, v jakém leží na hřebeni) vypište na výstup jeden řádek obsahující jediné celé číslo: její prominenci.

Příklady

Vstup:	*Výstup:*
`7`	`47`
`47 42 47 42 47 47 42`	`47`
	`47`

V tomto pohoří jsou tři hory: na indexu 1, na indexu 3 a na indexech 5–6. Všechny tři jsou nejvyššími horami světa, a tedy mají prominenci rovnou své nadmořské výšce.

Vstup:	*Výstup:*
`7`	`5`
`47 42 47 43 48 48 42`	`4`
	`48`

Toto je podobné pohoří, ale tentokrát má jedinou nejvyšší horu světa, totiž masiv na indexech 5–6. Zbylé dvě hory proto mají výrazně menší prominenci než v předchozím příkladu.

Vstup:	*Výstup:*
7 47 43 47 42 48 48 42	5 5 48

Tentokrát mají hory na indexech 1 a 3 stejnou prominenci – pro obě platí, že pokud se z nich chceme dostat někam výš, je třeba přejet přes sedlo na indexu 4 s nadmořskou výškou 42.

Vstup:	*Výstup:*
10 1955 1840 1937 1837 2004 2024 1232 1660 1190 1946	118 97 2024 428 756

Zjednodušený kus Nízkých Tater. Postupně zleva doprava Chabenec (hora), sedlo pod Chabencem, Kotliská (hora), sedlo Poľany, Dereše, Chopok (hora), Čertovica, Homôľka (hora), sedlo Priehyba a Kráľova hoľa (hora).

Bodování

Ve všech vstupech platí $1 \leq n \leq 200\,000$ a $1 \leq h_i \leq 10^9$ pro každé $1 \leq i \leq n$. Následující tabulka popisuje jednotlivé sady vstupů (ve sloupci n je vždy horní odhad na n ve všech vstupech v dané sadě):

body	n	*další vlastnosti*
1	20	Všechna h_i jsou navzájem různá.
1	20	
1	5 000	Všechna h_i jsou navzájem různá.
2	5 000	
2	200 000	Všechna h_i jsou navzájem různá.
3	200 000	

P-I-3 Střelec

Střelec je klasická šachová figurka. Pohybuje se tak, že ho posuneme v některém šikmém směru (po některé diagonále) o jedno nebo více políček. (Pokud jsou políčka šachovnice klasickým způsobem obarvena černou a bílou barvou, bude se střelec vždy pohybovat jen po políčkách jedné barvy.)

Máme obří šachovnici s r řádky a s sloupci, na jejímž každém políčku je napsané jedno písmeno. Chceme po ní přeskákat střelcem tak, aby se postupně zastavil přesně na písmenkách daného slova w. (Začít můžeme na libovolném políčku obsahujícím první písmeno w, každým dalším tahem se musíme přesunout na políčko obsahující následující písmeno slova w.) Kolika různými způsoby to můžeme udělat?

Jelikož správná odpověď může být opravdu velká, počítejte pouze její zbytek po dělení číslem $10^9 + 7$.

Formát vstupu

Na prvním řádku dostanete dvě mezerou oddělená celá čísla r a s. Na druhém řádku dostanete slovo w. Na i-tém z následujících r řádků dostanete řetězec délky s popisující, která písmena jsou na i-tém řádku šachovnice.

Formát výstupu

Vypište řádek obsahující jedno celé číslo, totiž zbytek počtu různých způsobů, jak vyskákat na šachovnici slovo w, po dělení číslem $10^9 + 7$.

Bodování

Můžete předpokládat, že všechna písmena na vstupu jsou velká písmena anglické abecedy. Speciálně tedy velikost abecedy můžete považovat za konstantu.

Plný počet bodů může získat řešení, které efektivně vyřeší vstupy s $r, s \leq 1000$ a $|w| \leq 100$.

Až 7 bodů může získat řešení, které efektivně vyřeší vstupy s $r, s, |w| \leq 100$.

Až 5 bodů může získat řešení, které efektivně vyřeší vstupy s $r, s \leq 6$ a $|w| \leq 12$.

Až 3 body může získat řešení, které efektivně vyřeší vstupy, v nichž je nejvíce 10^6 různých způsobů, jak po šachovnici vyskákat slovo w.

Příklady

Vstup:	*Výstup:*
3 4 MOP SUPI NOSI MOPY	2

Vstup:	*Výstup:*
3 5 OREL VASEK HRAJE SACHY	0

Jelikož na šachovnici není žádné O ani L, je zřejmé, že se slovo OREL poskládat nedá.

Vstup:	*Výstup:*
3 5 UUUUUU UUUUU UBAFU UUUUU	224

Pokud bychom řádky a sloupce číslovali od nuly zleva nahoře, jedna z přípustných možností skákání je $(1,0) \to (0,1) \to (2,3) \to (1,4) \to (2,3) \to (1,4)$.

Vstup:	*Výstup:*
6 8 KSP KSPPSKSP PPSKSKSP SKKSKSKP PPPSKSKK KKSKSKPP PPPSKSKP	102

Jedno přípustné řešení je začít na K úplně vlevo nahoře, odtud se posunout o tři políčka doprava dolů na S, a odtud o dvě políčka doleva dolů na P. Jiné přípustné řešení začne stejným pohybem na S a potom se vrátí o dvě políčka doleva nahoru na jiné P.

P-I-4 Vozidlo na Marsu

K této úloze se vztahuje studijní text uvedený na následujících stranách. Doporučujeme vám nejprve si přečíst studijní text a až potom se vrátit k samotným soutěžním úkolům.

a) *(2 body)* V lokalitě `jáma` jsou nějaké kameny. Napište libovolný program, po jehož provedení bude v lokalitě `jáma` čtyřikrát více kamenů, než na začátku výpočtu. (Nápověda: nejkratší řešení této podúlohy je tvořeno pouze dvěma příkazy.)

b) *(2 body)*

V lokalitách `A` a `B` máme nějaké neznámé počty kamenů, které označíme a a b. Lokalita `C` je prázdná. Napište program, který porovná hodnoty a a b. Pokud jsou stejné, vozidlo uloží do lokality `C` jeden kámen, jinak ji nechá prázdnou. Po skončení výpočtu musí v lokalitách `A` a `B` zůstat zachovány původní počty kamenů.

c) *(3 body)*

Napište pro vozidlo makro na dělení se zbytkem. Toto makro bude mít jako parametry čtyři lokality: První dvě na začátku výpočtu obsahují nějaké neznámé počty kamenů, které označíme a a b (můžete předpokládat, že $b > 0$). Tyto počty kamenů v nich musí zůstat zachovány i po skončení výpočtu. Druhé dvě lokality jsou na začátku výpočtu prázdné a chceme do nich uložit celou část podílu ($\lfloor a/b \rfloor$) a zbytek po dělení ($a \bmod b$).

d) *(3 body)*

V lokalitách `A` a `B` máme nějaké neznámé počty kamenů, které označíme a a b. Můžete předpokládat, že obě tyto hodnoty jsou kladné. Lokalita `C` je prázdná. Napište program, který do lokality `C` uloží přesně $\mathrm{nsd}(a, b)$ kamenů (kde nsd označuje největšího společného dělitele). V této podúloze je povoleno libovolně změnit obsah lokalit `A` a `B`.

Studijní text

Na Mars jsme dopravili průzkumné terénní vozidlo. Měli jsme s ním velké plány, ale zasáhla ho písečná bouře a zničila mu skoro všechna periferní zařízení, takže vozidlo zůstalo téměř nepoužitelné. Nyní se může jenom přesouvat mezi různými lokalitami a vozit z místa na místo kameny. Naším úkolem bude naučit vozidlo provádět alespoň některé použitelné výpočty. **Nebude nám přitom záležet na časové složitosti** programů, pouze na jejich korektnosti.

Do vozidla můžeme na dálku nahrát program: konečnou posloupnost příkazů. Některé příkazy mohou mít návěstí (labels) – symbolická jména, pomocí nichž se na příkazy můžeme odkazovat.

Vozidlo zná na Marsu dvě speciální lokality: *jídelnu* a *kamenolom*. Pro jednoduchost je budeme značit `J` a `K`. V jídelně je momentálně právě jeden kámen, jinak je to lokalita jako každá jiná. V kamenolomu je vždy k dispozici dostatečné množství kamenů.

Ke každému jinému řetězci má vozidlo přiřazenu nějakou unikátní lokalitu na Marsu, kam je možné ukládat kameny. Není-li řečeno jinak, předpokládáme, že všechny tyto lokality jsou prázdné – neobsahují žádné kameny.

Vozidlo umí provádět jediný příkaz, který vypadá následovně:

1. Jeď do lokality `Y`. Spočítej si do pomocné proměnné, kolik je tam kamenů.
2. Jeď do lokality `X`. Zkus tam naložit tolik kamenů, kolik udává pomocná proměnná.
3. Pokud se to podařilo, odvez tyto kameny do lokality `Z`, tam je vysyp a pokračuj následujícím příkazem.
4. Pokud se to nepodařilo (tzn. v lokalitě `X` není dost kamenů), uveď lokalitu `X` do původního stavu a pokračuj příkazem s návěstím `N`.

Program, který nahrajeme do vozidla, bude tedy posloupností takovýchto příkazů. Příkaz včetně návěstí budeme zapisovat takto:

```
návěstí: přenes X Y Z N
```

Jednotlivé parametry tohoto příkazu můžeme číst následovně:

```
přenes (odkud) (kolik) (kam) (co dělat při neúspěchu)
```

Na místě čtvrtého parametru můžeme napsat `-` (pomlčku), jestliže chceme, aby výpočet programu i v případě neúspěchu pokračoval následujícím příkazem.

Za `X`, `Y` a `Z` můžete dosadit libovolné lokality, dokonce nemusí být navzájem různé. Jediným omezením je, že kamenolom (kde je „nekonečně mnoho“ kamenů) nesmíme použít jako lokalitu `Y`.

Výpočet programu skončí, když se dostane na neexistující příkaz – tedy buď provedeme poslední příkaz programu a máme pokračovat následujícím příkazem, nebo skočíme na neexistující návěstí.

Příklad 1: příkazy, které skoro nic nedělají

Co udělá vozidlo, když mu dáme příkaz `přenes X X X I`? Spočítá kameny v lokalitě `X`, potom je všechny naloží, na stejném místě je zase všechny vysype a bude pokračovat následujícím příkazem. Tímto příkazem tedy Mars vůbec nezměníme.

Co udělá vozidlo, když mu dáme příkaz `přenes X Y X I`? Také tentokrát se počet kamenů v žádné lokalitě nezmění, existují ale dva možné průběhy výpočtu: jestliže bylo v lokalitě `Y` více kamenů než v lokalitě `X`, výpočet bude pokračovat příkazem s návěstím `I`.

Příklad 2: vyprázdni lokalitu

Rozmyslete si sami, jakým příkazem můžeme z lokality odstranit všechny kameny.

Řešení: Pro vyprázdnění lokality `X` můžeme použít příkaz `přenes X X K -`. Vozidlo spočítá kameny v lokalitě `X`, všechny je naloží a vysype je v kamenolomu.

Příklad 3: naučíme vozidlo sčítat

V lokalitách `A` a `B` máme nějaké neznámé počty kamenů, lokalita `C` je prázdná. Chtěli bychom mít v lokalitě `C` tolik kamenů, kolik jich je v lokalitách `A` a `B` dohromady. Lokality `A` a `B` přitom chceme ponechat beze změny.

Dříve než si přečtete řešení, zkuste ho opět sami vymyslet.

Řešení: Stačí příslušné počty kamenů převézt z kamenolomu do lokality `C`. To zajistíme následujícím programem:

```
přenes K A C -
přenes K B C -
```

Příklad 4: naučíme vozidlo odčítat

V lokalitách `A` a `B` máme nějaké neznámé počty kamenů, které označíme a a b. Lokalita `C` je prázdná. Chtěli bychom mít v lokalitě `C` počet kamenů rovný $a - b$, resp. rovný nule, pokud $b > a$. Lokality `A` a `B` přitom chceme ponechat beze změny.

(Na rozdíl od příkazu `přenes`, který odebere kameny pouze tehdy, pokud může vzít celý požadovaný počet, při našem odčítání chceme odebrat vždy tolik kamenů, kolik je možné.)

Řešení: Do lokality `C` přivezeme z kamenolomu a kamenů a potom se pokusíme odebrat z nich b. Když se nám to podaří, jsme hotovi, jinak všechny kameny z lokality `C` odvezeme zpět do kamenolomu.

```
            přenes K A C -
            přenes C B K nulování
            přenes prázdná J K konec
nulování:   přenes C C K -
```

Všimněte si, že ve třetím kroku (který se provádí, jestliže jsme z lokality `C` úspěšně odebrali b kamenů) se pokusíme z prázdné lokality odvézt jeden kámen. To se nám zaručeně nepodaří, program proto skočí na neexistující návěstí „konec“ a tím výpočet skončí. Čtvrtý příkaz se tedy provede jenom tehdy, když na něj skočíme z druhého příkazu.

Příklad 5: naučíme vozidlo násobit

V lokalitách `A` a `B` máme nějaké neznámé počty kamenů, které označíme a a b. Lokalita `C` je prázdná. Chtěli bychom mít v lokalitě `C` počet kamenů rovný $a \cdot b$. Lokality `A` a `B` přitom chceme ponechat beze změny.

Toto už nedokážeme provést v konstantním čase. Násobení je ale jenom opakované sčítání: hromadu ab kamenů získáme tak, že na ni a-krát přivezeme b kamenů.

```
            přenes K A Akopie -
cyklus:     přenes Akopie J K konec
            přenes K B C -
            přenes prázdná J K cyklus
```

Nejprve si vytvoříme kopii lokality `A`, kterou potom během výpočtu zničíme. Dále opakujeme, dokud to jde: vezmi jeden kámen z lokality `Akopie` a přidej b kamenů do lokality `C`.

Makra

Při programování našeho průzkumného vozidla můžeme definovat makra: vezmeme nějakou posloupnost příkazů a přiřadíme jí symbolické jméno, abychom nemuseli tutéž posloupnost příkazů rozepisovat vícekrát. Makro může mít parametry – při různých použitích takového makra se budou provádět stejné příkazy, ale pro jiné lokality a jiná návěstí.

Uvnitř makra můžeme používat také pomocné lokality. Lokality, které zapíšeme v definici makra do hranatých závorek, budou při každém použití makra nahrazeny jinou lokalitou, která se nikde jinde v programu nepoužívá.

Totéž se automaticky stane i se všemi návěstími, která dáme příkazům v definici makra. Každé takové návěstí je tedy viditelné jen z jednoho konkrétního použití makra.

V definici makra můžeme používat i jiná makra, která jsme definovali dříve.

Ukážeme si nyní několik příkladů definice makra. Řádky začínající mřížkou obsahují komentáře.

```
# příkaz, který nic nezmění, pouze jeden krok čeká
MAKRO čekej
          přenes J J J -
KONEC

# makro se dvěma parametry - lokalitami:
# do Y přidá tolik kamenů, kolik jich je v X
MAKRO přidej X Y
          přenes K X Y -
KONEC

# makro s jedním parametrem - návěstím: skočí na dané návěstí
MAKRO skoč N
          přenes [nová_prázdná_lokalita] J K N
KONEC

# makro pro násobení: do Z přidá součin počtů kamenů v X a Y
MAKRO vynásob X Y Z
          přidej X [Xkopie]
  cyklus: přenes [Xkopie] J K konec
          přenes K Y Z -
          skoč cyklus
   konec: čekej
KONEC
```

Všimněte si, že když jsme původně psali samostatný program pro násobení, stačilo nám, že návěstí `konec` neexistuje a skokem na něj jsme ukončili program. Při psaní makra toto návěstí umístíme před prázdný příkaz na konci makra, neboť chceme, aby po ukončení násobení program pokračoval dále ve výpočtu.

Příklad 6: naučíme vozidlo počítat třetí mocninu

Pomocí výše uvedených maker snadno napíšeme program, který bude počítat třetí mocninu. V lokalitě `A` máme nějaký neznámý počet kamenů, který označíme a. Lokalita `B` je prázdná a chceme do ní dovézt a^3 kamenů.

```
        vynásob A A pomocná_lokalita
        vynásob A pomocná_lokalita B
```

Pro názornost ještě ukážeme, jak by vypadal stejný program, kdybychom všechna makra nahradili jejich definicemi.

```
        přenes K A Akopie1 -
cyklus1: přenes Akopie1 J K konec1
        přenes K A pomocná_lokalita  -
        přenes cokoliv1 J K cyklus1
 konec1: přenes J J J -
        přenes K A Akopie2 -
cyklus2: přenes Akopie2 J K konec2
        přenes K pomocná_lokalita B -
        přenes cokoliv2 J K cyklus2
 konec2: přenes J J J -
```