

Korespondenční Seminář z Programování

ZAČÁTEČNICKÁ KATEGORIE

38. ročník



KSP-Z

Červen 2026

Právě se díváte na leták páté série 38. ročníku **KSP-Z**, neboli **Korespondenčního Semináře z Programování, Začátečnické** kategorie. Zapojit se může **každý středoškolák i základoškolák**.

V průběhu roku vydáváme několik dalších sérií úloh podobných této. Pokud budete mít jakoukoliv otázku, neváhejte se zeptat. Kontaktní adresy najdete v patičce na konci letáku. Přejeme hodně štěstí!

Termín série: neděle 28. června ve 32:00 (tedy další ráno v 8:00), praktické úlohy za třetinu bodů až do 6. července

Obsah série: **3 praktické úlohy** (značené ) – K těmto úlohám je nutné napsat program (v libovolném vhodném jazyce), stáhnout si z našeho webu vstupní data a odevzdat odpovídající výstup.
1 teoretická úloha (značená ) – U této úlohy nás zajímá hlavně slovní popis řešení, ve kterém byste měli zdůvodnit jeho funkčnost a ideálně nás i přesvědčit o jeho efektivitě.

Odevzdávání: Přes web na adrese <https://ksp.mff.cuni.cz/z/odevzdavatko/>



Zadání páté série začátečnické kategorie 38. ročníku KSP

38-Z5-1 Hroší-LZW dekomprese 8 bodů

Kevin přišel do serverovny a zjistil, že Sára neudělala nejen zkomprimovanou zálohu, ale také zkomprimovala původní data. Teď ale nedokáže spustit svůj server, protože neumí s komprimovanými daty pracovat. Sára použila Hroší-LZW kompresi,¹ ale rozhodla se, že slovník zabírá zbytečný prostor, takže zkomprimovaný text uložila bez něj.

Naštěstí ale Hroší-LZW umí slovník rekonstruovat z komprimovaného textu, takže není potřeba mít slovník uložený někde jinde. Pomůžete Kevinovi dekomprimovat data na serveru?

Toto je praktická open-data úloha. V odevzdávacím systému si necháte vygenerovat vstupy a odevzdáte příslušné výstupy. Záleží jen na vás, jak výstupy vyrobíte.

Hroší-LZW dekomprese

Dekomprese začíná s posloupností kódů slov a částečným slovníkem obsahujícím kódy 0–25 odpovídající slovům o jednom znaku a–z. Aby uměla všechny kódy převést zpět na data, potřebuje celý slovník. K jeho rekonstrukci musí replikovat chování komprese.

Prozkoumejme, proč byl každý kód slova při kompresi zapsán: Pokud je to poslední kód, šlo o konec původního textu. Jinak komprese přečetla slovo $w + c$, které nebylo ve slovníku, avšak w mělo kód K : ten byl vypsán, čímž bylo zpracované w , a do slovníku přibylo slovo $w + c$ s novým kódem P . Znak c se pak stal prvním znakem dalšího slova.

Když tedy dekomprese přečte kód K , vypíše jemu odpovídající slovo w . V tuto chvíli, pokud nejde o poslední kód, má do slovníku přibýt slovo $w + c$ s kódem P . Přečtení K však c neodhalí, je třeba jej získat z následujícího kódu. Pro ten mohou nastat dvě situace:

- buď je to právě přidávaný kód P pro slovo $w + c$
- nebo je to dříve přidáný kód s jeho odpovídajícím slovem

V obou případech zjistíme c , takže $w + c$ přibude do slovníku s kódem P a dekomprese pokračuje dalším kódem.

Formát vstupu: Vstup má vždy jeden řádek, který obsahuje několik čísel oddělených mezerami, která představují komprimovaný text.

Slibujeme, že vstup je validní výstup Hroší-LZW komprese z předchozí úlohy bez slovníku, ten si musíte rekonstruovat sami.

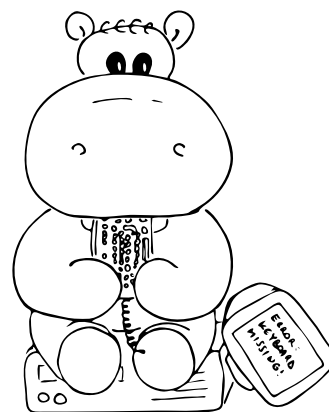
Formát výstupu: Vypište jediný řádek – dekomprimovaný text složený z malých písmen anglické abecedy.

Ukázkový vstup: *Ukázkový výstup:*

2 1 0 28 3 28 cbaadaa

Vysvětlení chování pro každý kód:

- 2 je kód c , tedy vypíšeme $w := c$.
- 1 je kód b , tedy přidáme kód 26 patřící slovu $w + c = cb$ a vypíšeme $w := b$.
- 0 je kód a , tedy kód 27 patří ba a vypíšeme $w := a$.
- 28 je přidávaný kód, tedy $w + c = aa$ je jeho slovo, které přidáme a vypíšeme.
- 3 je kód d , tedy kód 29 patří aad a vypíšeme $w := d$.
- 28 je kód aa , tedy kód 30 patří da a vypíšeme $w := aa$.
- Další kód není, takže končíme.



38-Z5-2 Rovnice 10 bodů

Kevin se těší na konec školy, ale předtím, než se ho dočká, musí ještě vyřešit poslední úkol z matiky. Myslel si, že to bude jednoduchý, rychlý dvacetiminutový úkol, přeci jen měl pouze šest otázek. Když ale otevřel zadání, zjistil, že každá otázka má několik podotázek. Ba, víc než několik. Poslední tři otázky měly tolik podotázek, že to nemohl ani spočítat.

¹ S tou jste se mohli setkat v předchozí sérii.

Všechny otázky a podotázky byly na řešení soustav lineárních rovnic, přesněji, řešení soustav tří rovnic o třech neznámých. Kevin si pamatuje, že učitelka povolila libovolný způsob řešení, tak si Kevin vybral ten programátorský. Místo toho, aby je počítal ručně, chce napsat program, který mu je vyřeší. Bohužel neví, jak se s tím vypořádat. Pomůžete mu?

Toto je praktická open-data úloha. V odevzdávacím systému si necháte vygenerovat vstupy a odevzdáte příslušné výstupy. Záleží jen na vás, jak výstupy vyrobíte.

Formát vstupu: Na první řádku vstupu dostanete číslo N – počet soustav rovnic, které jsou v daném vstupu. Na následujících $4N$ řádcích jsou pak jednotlivé soustavy rovnic, oddělené prázdným řádkem.

Každý řádek bude ve formátu $ax + by + cz = d$, kde a, b, c a d jsou celá čísla a x, y a z jsou neznámé. Neznámé mohou být libovolná malá písmenka z anglické abecedy a nemusí být ve stejném pořadí v rámci soustavy rovnic.

Slibujeme, že koeficienty a, b, c a d budou poměrně malé a vždy nenulové. Podle způsobu, jak budete řešit soustavy, se vám nemusí vejít mezivýsledky do 32-bitových celých čísel, ale určitě se vejdou do 64-bitových celých čísel.

Formát výstupu: Pro každou soustavu rovnic vypište na samostatný řádek proměnné a jejich hodnoty ve formátu $x = 1y = 2z = 3$. Nezáleží na pořadí proměnných, ale musí být oddělené mezerou.

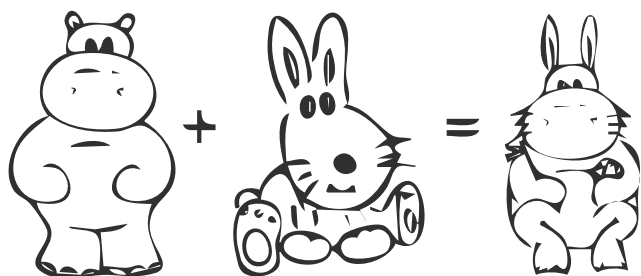
Ukázkový vstup:

```
2
-1y - 5c + 4e = -2
4c - 2y - 5e = -16
-3e + 3y + 4c = 13

5v + 3s + 5x = 40
-2v - 5x - 5s = -47
3s + 2x + 1v = 24
```

Ukázkový výstup:

```
c=1 e=2 y=5
s=5 v=1 x=4
```



38-Z5-3 Už žádné zbytečné kopce! 12 bodů

Sára chce jet v příštím školním roce na výměnný pobyt do zahraničí. Zrovna si vybírá svou vysněnou destinaci. Před pár měsíci by Sáru zajímala všemožná kritéria: dá se tam najít levné bydlení? Jakou reputaci má tamní škola? Jsou v okolí zajímavá místa na výlety? Na jaře ale začala s Petrem jezdit na kole a chytlo ji to tak moc, že se teď všude pohybuje prakticky jen na kole. Jenže Sářino městečko je samý kopec a neumíte si představit, jak je to frustrující, když s vypětím všech sil vyšlápává kopec a přitom ví, že ho hned v zápětí bude zase muset sjet!

Rozhodla se proto, že takovou věc v zahraničí nechce podstupovat. Chce proto bydlet jen v takovém městě, které je *monotónní*: mezi každými dvěma místy v něm existuje cesta, která jen stoupá, nebo jen klesá. Už žádné zbytečné kopce!

Město, o kterém Sára uvažuje, si můžeme představit jako 2D mřížku o $R \times S$ políčkách. Pro každé políčko známe jeho nadmořskou výšku, která je navíc *pro všechna políčka unikátní*. Z každého políčka se můžeme přesunout na jeho čtyři sousedy (popř. méně, pokud je to políčko krajní). Zajímá nás, zda je město monotónní, tedy pro libovolná dvě políčka A a B platí, že existuje cesta z A do B , která jen klesá, popř. jen stoupá.

Toto je praktická open-data úloha. V odevzdávacím systému si necháte vygenerovat vstupy a odevzdáte příslušné výstupy. Záleží jen na vás, jak výstupy vyrobíte.

Formát vstupu: Na prvním řádku je číslo M – počet měst, o kterých Sára uvažuje. Následuje M popisů měst, všechny za sebou v jednom souboru. Každý popis města vypadá následovně: Na prvním řádku jsou dvě čísla R, S značící počet řádků a sloupců mapy popisující město. Na dalších R řádcích je popis 2D výškové mapy města. Slibujeme, že nadmořské výšky v rámci jednoho města jsou unikátní.

Formát výstupu: Vypište M řádků. Na i -tém řádku vypište informaci o i -tém městě. Pokud je i -té město monotónní, vypište ANO. Pokud není monotónní, vypište NE $x y$, kde x a y jsou nadmořské výšky dvou políček, mezi kterými neexistuje cesta, která jen stoupá, nebo jen klesá.

Ukázkový vstup:

```
2
3 3
1 3 15
7 5 13
9 10 12
3 3
1 2 3
4 5 6
7 8 9
```

Ukázkový výstup:

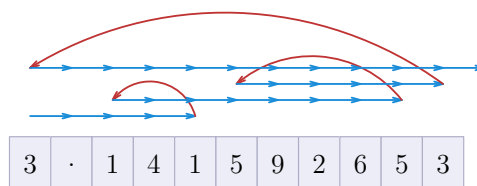
```
ANO
NE 3 4
```

38-Z5-4 Gramofon 14 bodů

Kevin zase jednou zašel na garážový výprodej, kde ho tentokrát zaujal zachovalý gramofon za výhodnou cenu. Přikoupil si k němu pár desek a po návratu domů se ho hned jal vyzkoušet. Co ale neslyšel! Jako první si pustil nahrávku s jeho oblíbeným číslem π . Místo líbezného „3.141592653...“ však zazněl následující horor:

„3.141415926592653.141592653...“.

Z toho se Kevinovi značně zamotala hlava. Takovou fraškou nepůjde ani aproximovat gravitační zrychlení na Zemi! (Věděli jste, že $\pi^2 \approx g$?) Po chvíli poslouchání mu došlo, co se s nahrávkou stalo. Pokaždé, když měla zaznít číslice, kterou už v nahrávce slyšel, nahrávka se vrátila na předchozí výskyt této číslice a úsek odtud až po aktuální výskyt číslice *jednou* zopakovala. Průchod číslem π je vidět na následujícím obrázku:



Kevin by nyní pro různé nahrávky zajímalo, kolikrát každá číslice na takto rozbitém gramofonu zazní. Pomůžete mu to zjistit?

Na vstupu máte posloupnost N čísel v rozsahu $1 \dots K$. (Ano, našich „číslí“ je K – tedy to mohou být v desítkovém zápisu víceciferná čísla.) Pro každou z K číslic vypište, kolikrát celkově bude na takto rozbitém gramofonu přehrána.

Toto je teoretická úloha. Není nutné ji programovat, odevzdává se pouze slovní popis algoritmu. Více informací zde: <http://ksp.mff.cuni.cz/viz/tinfo>

