1

Program přečte dvě celá čísla a vypíše jejich celočíselný podíl. Pokud tato čísla nejdou vydělit (rozmyslete si, kdy není dělení definováno), vypíše "NELZE". Např. pro 9 a 4 je tedy výsledek 2.

2

Na vstupu dostanete posloupnost celých čísel ukončenou číslem -1, které se do posloupnosti už nepočítá.

Napište program, který vytiskne hodnotu druhého největšího prvku této posloupnosti.  
Můžete počítat s tím, že posloupnost je dostatečně dlouhá.

*Příklad 1:*

*Vstup:*

1 2 3 4 5 -1

*Odpovídající výstup:*

4

*Příklad 2:*

*Vstup:*

1 2 3 7 7 -1

*Odpovídající výstup:*

7

3

Je dána posloupnost N kladných celých čísel, kde N<=1000. Nalezněte největší číslo obsažené v posloupnosti a vypište pozice všech jeho výskytů.  
  
**Vstup:** První řádek vstupu obsahuje počet prvků posloupnosti N (1 <= N <= 1000). Druhý řádek obsahuje N prvků posloupnosti oddělených mezerami.  
  
**Výstup:** Na prvním řádku vypište největší číslo obsažené v posloupnosti. Na druhém řádku vypište pozice všech jeho výskytů oddělené mezerou.   
  
**Příklad:**  
*vstup:*  
7  
568 1 546 84 568 215 546  
  
*výstup:*  
568  
1 5

4

Číslo nazveme perfektním, pokud je součtem svých (vlastních) dělitelů, tedy například 6 = 1 + 2 + 3, oproti tomu číslo 36 perfektní není, protože součet jeho dělitelů je 1 + 2 + 3 + 4 + 6 + 12 + 18 = 46. Číslo nazveme čtvercem, pokud je druhou mocninou nějakého celého čísla. Podobně číslo nazveme krychlí, pokud je třetí mocninou nějakého celého čísla. Napište program, který vyšetřuje tyto tři vlastnosti a oznámí je. Je-li číslo perfektní, vypíše 'P', je-li číslo čtverec, vypíše 'C' a pro krychli 'K' v tomto pořadí. Kupř. na vstup 64 tedy bude odpověď 'CK'.

5

Máme osít N záhonů. Na každém záhonu může být buď mrkev nebo petržel, avšak nikdy nesmí být žádné dva sousední záhony osety petrželí. Sestavte program, který ze standardního vstupu přečte počet záhonů N a na standardní výstup vypíše číslo P, kde P je počet různých přípustných osetí.

Snažte se o efektivní výpočet. Vstupní číslo N nesmí být větší než 44. Pro výpočty použijte typ longint.

**Příklad:**

*Vstup:*

3

*výstup:*

5

*Vysvětlení:* Pro tři záhony existuje 5 způsobů, jak je osít.

6

Navrhněte funkci sloužící k testování, zda je dekadický zápis zadaného přirozeného čísla symetrický (tj. stejný při čtení zleva i zprava).

Program dostane na vstupu posloupnost přirozených čísel ukončenou 0, která do posloupnosti nepatří. Každý prvek posloupnosti bude na jednom řádku. Na výstup vypíše ta čísla ze vstupní posloupnosti, jejichž zápis je symetrický. Čísla na výstupu budou oddělená mezerou.

Pokud vstupní číslo obsahuje na konci nuly, pak symetrický zápis nemá. Všechna čísla se vejdou do 32-bitového integeru.

*Příklad:*  
  
*Vstup*  
  
15651  
8521  
3647  
123321  
51215  
3000  
0  
  
*Výstup*  
  
15651 123321 51215

7

Určete poslední dvě číslice zadané mocniny. Na prvním řádku vstupu je číslo **a,** které je základem mocniny. Na druhém řádku je zadaný exponent **b.** Zatímco základ je zadán v desítkové soustavě, exponent je zadán v soustavě dvojkové. Určete poslední dvojčíslí mocniny **ab.** Povšimněte si, prosím, že neříkáme, zda se zadaná čísla vejdou vůbec do nějakého datového typu (neřkuli do paměti).

**Příklad:**  
**Vstup:**  
111111111111111111111111111111111111111111111111111111111111111111  
111111111111111111111111111111111111111111111111111111111111111111  
**Výstup:**  
31

8

Na standardním vstupu jsou zadána tři kladná celá čísla D, M, R oddělená mezerami. Tato čísla představují datum (den, měsíc, rok) - jedná se o korektní datum z rozsahu platnosti současného Gregoriánského kalendáře. Program určí a vypíše na standardní výstup datum následujícího dne, a to opět ve tvaru tří celých čísel oddělených mezerami.

Poznámka: Jednotlivé měsíce v roce mají po řadě 31, 28 (nebo 29), 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 dní. Rok je přestupný, je-li jeho letopočet dělitelný 4, s výjimkou celých století, která přestupná nejsou. Pokud je ovšem číslo roku dělitelné 400, pak je tento rok přestupný, i když se jedná o celé století.

Příklad vstupu:   
30 4 2009

Odpovídající výstup:   
1 5 2009

9

Naprogramujte, jak se maji prehazovat kotouce Ha-noiskych vezi. Na vstupu prectete cislo n urcujici pocet kotoucu. Vypiste posloupnost pokynu, jak prestehovat n kotoucu z prvni tyce na tyc druhou. Instrukce piste stylem "Kotouc cislo\_kotouce z cislo\_tyce na cislo\_tyce", kde cislo\_kotouce a poobakrat cislo\_tyce jsou cisla. Kazdou instrukci piste na zvlastni radek. Tedy napriklad na vstup 2 vypisete tri radky a to: "Kotouc 1 z 1 na 3" "Kotouc 2 z 1 na 2" a "Kotouc 1 z 3 na 2". Predpokladejte, ze vstup se vejde do 32bitoveho integeru.

10

Permutací *N*-prvkové množiny přirozených čísel {1, 2, ..., *N*} rozumíme každé uspořádání prvků této množiny. Uvedená množina má *N!* různých permutací. Všechny permutace množiny můžeme uspořádat lexikografickým uspořádáním, tzn. takovým způsobem, jak se řadí hesla v slovníku. O pořadí dvou zvolených permutací rozhoduje nejprve jejich první prvek, v případě shody prvního prvku pak druhý prvek, atd. První permutací v lexikografickém uspořádání je (1, 2, ...,*N*), poslední permutací je (*N*, *N*-1, ..., 2, 1).

Na prvním řádku standardního vstupu je jedno kladné celé číslo *N*, které není větší než 100. Na druhém řádku následuje nějaká permutace množiny přirozených čísel {1, 2, ..., *N*}, přičemž jednotlivá čísla jsou na vstupu od sebe oddělena mezerami.

Program k zadané permutaci určí permutaci bezprostředně po ní následující v lexikografickém uspořádání. Výslednou permutaci vypíše do jednoho řádku na standardní výstup, její jednotlivé členy budou na výstupu opět odděleny mezerami. Pokud byla na vstupu zadána poslední permutace v lexikogafickém uspořádání a následující permutace tedy neexistuje, program vypíše pouze znakový řetězec **NEEXISTUJE**

*Příklad vstupu:*

6

1 3 6 2 5 4

*Odpovídající výstup:*

1 3 6 4 2 5

11

**Mřížková šifra**  
Tato šifra vznikla v Egyptě, poté co byla po dlouhém a usilovném zkoumání prolomena slavná Caesarova šifra. Tato šifra je mnohem komplikovanější a říká se jí šifra mřížková.

Mřížková šifra nepoužívá jako klíč jedno číslo, ale čtvercovou hliněnou destičku. Tato destička je rozdělená na N·N malých čtverečků, z nichž některé jsou děravé. Kryptografikové této destičce přezdívají mřížka.

Šifrování probíhá tak, že se mřížka přiloží k papyru, postupuje se po řádcích a do každé dírky se vepíše jeden hieroglyf. Poté se mřížka otočí o 90° doprava a postup se opakuje. Celkem 4\* zapíšeme do všech dírek a nakonec mřížku z papíru sundáme. Nyní, pokud byla mřížka v pořádku, tak koukáme na papyrus, na němž se nachází matice N·N písmen. Tuto matici poté přečteme po řádcích a máme zašifrovaný text.

**Problém s dodavatelem**  
Jistě si všimnete, že ne každá proděravěná destička je funkční mřížka. Pokud něco není v pořádku, může se stát, že na jednom místě na papyru skončí více než jeden hieroglyf nebo že někde zbudou volná místa.

Problém je, že dodavatel destiček je obchodník jak se sluší a patří. Prodal by cokoliv, jen aby dostal zaplaceno. Je tedy třeba napřed celou dodávku zkontrolovat a v případě, že obsahuje padělky, hodit dodavatele posvátným krokodýlům.

**Vstup**  
Vstup je v souboru dodavka.in. Na prvním řádku se nachází číslo D, které udává počet destiček v této dodávce. Pod tím následuje D bloků, každý popisující jednu destičku.

Každý blok má na svém prvním řádku číslo N (N ≤ 255), rozměr destičky. Pod ním následuje N řádků, každý má N znaků a popisují destičku. Znak @ značí dírku, zatímco znak # říká, že zde žádná dírka není.

Například:  
*dodavka.in*  
3  
2  
@#  
##  
4  
@##@  
####  
####  
@##@  
6  
@#####  
####@#  
####@#  
@#@#@#  
######  
#@@#@#

**Výstup**  
Výstup je očekáván v souboru krokodyl.out (out, přestože na výsledku záleží, jestli se něco dostane do krokodýla, nikoliv ven). Na prvních D řádcích jsou výsledky rozhodování o jednotlivých destičkách. Na posledním řádku bude celkový výsledek.

Výsledek každého rozhodování bude řetězec „mrizka“, pokud zadaná destička je funkční mřížka, nebo dvě celá čísla, M a Z. M je počet políček čtverce na papyru, kam se zapsalo mockrát (vícekrát, než jednou) a Z je počet políček, kde není žádný hieroglyf.

Celkový výsledek je buď řetězec „koupit!“, pokud jsou všechny destičky v pořádku, nebo „krokodyli se nazerou!“, pokud je alespoň jedna destička padělek.

*krokodyl.out*  
mrizka  
4 12  
mrizka  
krokodyli se nazerou!

12

Program bude hledat nejkratší cestu šachovým králem na šachovnici 8x8, kde na některá políčka nelze vstoupit.

Vstup programu obsahuje popořadě:

* počet překážek
* souřadnice jednotlivých překážek (dvojice čísel v rozsahu 1..8)
* souřadnice výchozího políčka
* souřadnice cílového políčka

Počet překážek je zapsán na samostatném řádku, souřadnice jsou vždy zapsány jako dvojice čísel na jednom řádku oddělená mezerou.

Výstup je buď -1, pokud král na cílové políčko nemůže dojít NEBO počet kroků, které musí vykonat.

*Příklad vstupu:*

1

2 1

1 1

2 2

*Odpovídající výstup:*

1

13

Napište program, který vyhodnotí zadaní výraz v polské reverzní (prefixové) notaci, případně napíše, že je výraz chybný. Vstupem programu je právě jeden řádek se zadaným výrazem. Výraz se skládá z operátorů '+', '-', '\*' a '/' a celých čísel. Všechny operátory i čísla jsou navzájem odděleny mezerami. Můžete předpokládat, že všechna čísla včetně všech mezivýsledků se vejdou do datového typu longint. Operátor '/' představuje celočíselné dělení. Pokud by mělo dojít k dělení nulou, vypište, že se jedná o chybný výraz. Výstupem programu je jedno celé číslo, které představuje hodnotu výrazu (nebo slovo "CHYBA". Příklad 1: Vstup: / + 1 2 2 Výstup: 1

14

Na standardním vstupu je zadáno číslo *N* (1 ≤ *N* ≤ 40). Vypište na standardní výstup všechny možnosti, jak toto číslo rozložit na součet celých kladných čísel. Každý rozklad musí být uveden na samostatném řádku, sčítanci vyjmenováni od nejmenšího k největšímu a odděleny znaménkem "+". Na pořadí řádků nezáleží.

Například pro *N=5* je jeden ze správných výstupů následující:

1+1+1+1+1

1+1+1+2

1+1+3

1+2+2

1+4

2+3

5

15

Napište program, který zjistí a vypíše počet všech možných korektních uzávorkování výrazu pomocí *N* párů závorek (). Například pro *N*=3 existuje 5 takových možných uzávorkování:   
()()()   
(())()   
()(())   
(()())   
((()))   
Na standardním vstupu program dostane jedno celé číslo *N* určující počet párů závorek, výsledek výpočtu (opět jedno celé číslo) program vypíše na standardní výstup.

16

Napište program, který pro zadanou permutaci určí její pořadové číslo v lexikograficky uspořádaném seznamu všech permutací čísel 1 až N.  
  
Na prvním řádku vstupu program dostane číslo N. Na druhém řádku pak permutaci čísel 1 až N. Výstup tvoří jedno číslo: pořadí této permutace ve výše zmíněném seznamu. Můžete předpokládat, že výsledná hodnota se vždy vejde do datového typu longint.  
  
Příklad:  
Vstup:  
5  
1 4 2 3 5  
  
Výstup:  
13

17

Ze standardního vstupu přečtěte 4 kladná celá čísla m, n, sx, sy oddělená mezerami, která budou splňovat podmínky m < 20, n < 20, sx <= m, sy <= n.   
  
Startovní pole jezdce je dáno souřadnicemi [sx, sy], rozměry šachovnice jsou m sloupců krát n řádků.   
  
Úkolem je odpovědět na otázku, zda existuje posloupnost m\*n-1 tahů jezdcem takových, že na žádné pole šachovnice jezdec nevstoupí dvakrát. Požadovaný výstup (na stdout) je řetězec ANO nebo NE.   
  
Příklad vstupu: 3 3 2 1   
Správný výstup: NE   
  
Příklad vstupu: 5 5 2 1   
Správný výstup: NE   
  
Příklad vstupu: 10 3 1 1   
Správný výstup: ANO

18

Čtěte vstupní textový soubor text.in a určete, kolikrát se v něm vyskytuje jeho dvacáté slovo.

Slovem zde rozumíme libovolnou souvislou posloupnost písmen, pokud znak, který jí v souboru bezprostředně předchází, ani znak, který ji v souboru bezprostředně následuje, nejsou písmena. Za písmena zde považujeme pouze znaky anglické abecedy. Při porovnávání dvou slov nebereme do úvahy rozdíl mezi malým a velkým písmenem. Předpokládejte, že žádné slovo ve vstupním souboru nebude delší než 30 písmen.

Výsledek bude tedy celé číslo, které zapište na standardní výstup. Jestliže vstupní soubor obsahuje méně než 20 slov, správný výsledek je 0.

Požadavek: Vstupní soubor čtěte pouze jednou.

Příklad vstupního souboru:

A computer file is a block of arbitrary information, or resource for storing information, which is available to a computer program and is usually based on some kind of durable storage. A file is durable in the sense that it remains available for programs to use after the current program has finished. Computer files can be considered as the modern counterpart of paper documents which traditionally are kept in offices' and libraries' files, and this is the source of the term.

Dvacáté slovo v uvedeném souboru je slovo "computer", které se v celém textu vyskytuje třikrát. Správný výstup je tudíž číslo 3.

Jiný příklad vstupního souboru:

<p> Čtěte vstupní textový soubor <code>text.in</code> a určete, kolikrát se v něm případně vyskytuje jeho dvacáté slovo.   
</p>   
<p> Slovem zde rozumíme libovolnou souvislou posloupnost písmen, pokud znak, který jí v souboru bezprostředně předchází, ani znak, který ji v souboru bezprostředně následuje, nejsou písmena. Za písmena zde považujeme pouze znaky anglické abecedy. Při porovnávání dvou slov nebereme do úvahy rozdíl mezi malým a velkým písmenem. Předpokládejte, že žádné slovo ve vstupním souboru nebude delší než 30 písmen.   
</p>   
<p> Výsledek bude tedy celé číslo, které zapište na standardní výstup. Jestliže vstupní soubor obsahuje méně než 20 slov, správný výsledek je 0.   
</p>

Správný výstup pro uvedený text je číslo 15.

19

Při dobývání Tróje byla hrstka obránců zahnána řeckými válečníky do polorozbořené budovy chrámu a obklíčena. Aby unikli ponižující porážce, rozhodli se Trójané spáchat sebevraždu. Stoupli si do kruhu, přiřadili si čísla a ob jednoho postupně nalehli na svůj meč. Začali vynecháním vojáka s číslem 1 a pokračovali v kruhu tak dlouho, dokud nebyli všichni mrtví. Jeden z Trójanů, říkejme mu Josef, ale s tímto postupem nesouhlasil a mnohem radši by se vzdal Řekům a zachránil si život. Poraďte Josefovi, na jaké místo v kruhu si má stoupnout, aby při rozpočítávání zůstal jako poslední a nikdo z Trójanů se tak nedozvěděl o jeho potupě.

Na standardním vstupu dostanete jedno celé číslo N v rozsahu typu *longint* - počet trójských vojáků. Vypište na standardní výstup celé číslo J(N) - místo v kruhu, na které si má Josef stoupnout. Při nekorektním vstupu vypište pouze řetězec 'ERROR'.

Pro N = 5 probíhá tedy vyřazování takto:

1. 1 2 3 4 5 => 1 3 4 5 (vypadl voják 2)
2. 1 3 4 5 => 1 3 5 (vypadl voják 4)
3. 1 3 5 => 3 5 (vypadl voják 1)
4. 3 5 => 3 (vypadl voják 5)
5. J(N) = 3

*Příklad 1:*

*Vstup:*

37

*Odpovídající výstup:*

11

*Příklad 2:*

*Vstup:*

-5

*Odpovídající výstup:*

ERROR

20

Na standardním vstupu jsou zadána dvě kladná celá čísla N, M z rozmezí od 1 do 1000, čísla jsou na vstupu oddělena mezerou. Spočítejte přesně jejich podíl N/M a vyjádřete ho ve tvaru desetinného čísla s vyznačením případné periody. Výsledek vypište na standardní výstup.

Výsledek musí být uveden přesně v některém z následujících tvarů:

1. Pokud je podíl celočíselný, má výsledek podobu celočíselné konstanty (bez desetinné tečky!). Například pro N=100, M=5 bude výsledkem hodnota **20**

2. Pokud podíl není celočíselný, ale má konečný desetinný rozvoj, bude výsledkem přesně spočítané desetinné číslo obsahující ve svém zápisu desetinnou tečku. Například pro N=184, M=5 bude výsledkem hodnota **36.8**

3. Má-li podíl nekonečný desetinný rozvoj, je spočítán přesně a perioda je v něm vyznačena uzařením do závorek. Přitom se požaduje vždy nejkratší možný zápis výsledku, tzn. perioda musí být vyjádřena nejkratším způsobem a také předperioda musí být uvedena v nejkratším možném tvaru. Například pro N=1, M=3 bude jediným správným výsledkem hodnota **0.(3)** - není tedy možné uvést výsledek ve tvaru 0.(33), i když je to teoreticky také správná hodnota. Podobně pro N=100, M=7 bude jediným správným výsledkem hodnota **14.(285714)** - není povoleno zapsat výsledek třeba v podobě 14.2(857142). Pro N=11, M=24 je očekáván výsledek **0.458(3)**

21

Program najde nejdelší řadu, jakou lze postavit ze zadaných kostek domina a vytiskne její délku. Kostky lze při připojování otočit.

Kostky domina mají svá políčka ohodnocená hodnotami v rozsahu 1..38, počet kostek je nejvýše 16.

Vstupem programu je počet kostek N (nejvýše 16) a potom N dvojic čísel z rozsahu 1..38 popisující jednotlivé kostičky.

Například pro vstup:

5 1 2 1 2 2 3 2 17 2 17

Bude odpovídající výstup:

5

Tzn. nejdelší řada, kterou ze zadaných kostek můžeme sestavit (například 2-1 1-2 2-17 17-2 2-3), má délku 5.