double d = b[j];double d = b[j];b 1. Zapište **program**, který bude zjišťovat **nejdelší setříděnou podposloupnost** zadané číselné posloupnosti. Program má na výstupu vypsat:

a) délku nejdelší setříděné podposloupnosti,

b) index prvního prvku nejdelší setříděné podposloupnosti,

**Specifikace vstupu**

Program má umožnit při jednom spuštění zadání libovolného počtu posloupností. Každá posloupnost bude zadána počtem prvků a poté jednotlivými prvky. Program má končit v případě, že zadaná délka posloupnosti je menší než 1.

**Ukázka komunikace programu s uživatelem**

```
Zadej pocet:
7
Zadej posloupnost:
12 18 17 19 25 34 31
Nejdelsi setridena část delky 4
Zacatek 3. prvek

Zadej počet:
12
Zadej posloupnost:
12 18 45 76 48 42 7 10 13 19 21 26
Nejdelsi setridena cast delky 6
Zacatek 7. prvek

Zadej pocet:
0
```

2. Zapište **program**, který bude načítat dvojice číselných posloupností kladných celých čísel včetně 0 a zjistí **nejdelší společný úsek každé zadané dvojice posloupností**. Program má pro každou zadanou dvojici posloupností vypsat pouze délku nejdelšího společného úseku.

**Specifikace vstupu**

Program má umožnit při jednom spuštění zadání libovolného počtu zpracovávaných dvojic posloupností. Před načítáním další dvojice nechť program vypíše dotaz uživateli, zda pokračovat ve zpracování či nikoli – odpověď uživatele bude znak `'a'` nebo `'n'` (malými nebo velkými písmeny). Program má skončit v případě, že odpověď uživatele je `'n'`. Každá zadávaná posloupnost bude ukončena zadáním záporné hodnoty (záporná hodnota není součástí příslušné posloupnosti).

**Ukázka komunikace programu s uživatelem**

```
Pokracovat ve zpracovani (a/n):
a
Zadej první posloupnost:
0 1 0 1 2 4 8 12 65 0 11 6 -1
Zadej druhou posloupnost:
8 12 65 4 0 1 0 1 2 78 5 -1
Nejdelsi spolecny usek obou posloupnosti ma delku 5

Pokracovat ve zpracovani (a/n):
y
Pokracovat ve zpracovani (a/n):
Y
Pokracovat ve zpracovani (a/n):
A
```

```
Zadej první posloupnost:
4 1 2 3 1 2 12 6 8 5 0 1 0 -1
Zadej druhou posloupnost:
7 0 7 5 25 21 1 2 12 6 8 5 1 2 3 1 11 -1
Nejdelsi spolecny usek obou posloupnosti ma delku 6

Pokracovat ve zpracovani (a/n):
N
```

3. Zapište **program**, který pro každou dvojici načtených číselných posloupností určí, zda **první posloupnost obsahuje druhou posloupnost**.

   **Specifikace vstupu**

   Program má umožnit při jednom spuštění zadání libovolného počtu dvojic posloupností. Před načítáním další dvojice posloupností nechť program vypíše dotaz uživateli, zda pokračovat ve zpracování či nikoli – odpověď uživatele bude znak `'a'` nebo `'n'` (malými nebo velkými písmeny). Program má skončit v případě, že odpověď uživatele je `'n'`. Každá zadávaná posloupnost bude ukončena zadáním záporné hodnoty (záporná hodnota není součástí příslušné posloupnosti).

   **Ukázka komunikace programu s uživatelem**

```
Pokracovat ve zpracovani (a/n):
a
Zadej posloupnost:
0 1 0 1 2 4 8 12 65 1 2 0 1 1 6 -1
Hledana sekvence:
0 1 2 -1
Sekvence nalezena na pozici 3

Pokracovat ve zpracovani (a/n):
y
Pokracovat ve zpracovani (a/n):
Y
Pokracovat ve zpracovani (a/n):
A
Zadej posloupnost:
7 0 7 5 25 21 1 2 2 1 11 12 6 8 5 1 2 3 1 11 -1
Hledana sekvence:
1 2 3 2 1 -1
Sekvence nenalezena

Pokracovat ve zpracovani (a/n):
N
```

4. Zapište **program**, který bude načítat trojice vzestupně setříděných posloupností kladných čísel a **z každé trojice posloupností vytvoří jedinou setříděnou posloupnost**. Při vytváření výsledné posloupnosti využijte setříděnosti vstupních posloupností.

   **Specifikace vstupu**

   Program má umožnit při jednom spuštění zpracování libovolného počtu trojic posloupností. Před načítáním další trojice posloupností nechť program vypíše dotaz uživateli, zda pokračovat ve zpracování či nikoli – odpověď uživatele bude znak `'a'` nebo `'n'` (malými nebo velkými písmeny). Program má skončit v případě, že odpověď uživatele je `'n'`. Každá zadávaná posloupnost bude ukončena zadáním záporné hodnoty (záporná hodnota není součástí příslušné posloupnosti).

   **Ukázka komunikace programu s uživatelem**

```
Pokracovat ve zpracovani (a/n):
a
Zadej prvni posloupnost:
12 14 18 25 -1
Zadej druhou posloupnost:
4 8 65 124 128 -1
Zadej treti posloupnost:
8 10 20 77 -1
Vysledna posloupnost
4 8 8 10 12 14 18 20 25 65 77 124 128

Pokracovat ve zpracovani (a/n):
y
Pokracovat ve zpracovani (a/n):
Y
Pokracovat ve zpracovani (a/n):
N
```

5. Zapište **program**, který **do vzestupně setříděné posloupnosti kladných čísel bude zařazovat postupně další prvky** a to pouze v případě, že se příslušný prvek ve vytvářené posloupnosti doposud nevyskytuje. Při vyhledání pozice pro vložení dalšího prvku využijte setříděnosti posloupnosti – aplikujte binární vyhledávání.

   **Specifikace vstupu**

   Program má umožnit při jednom spuštění zpracování libovolného počtu zadání. Před zadáním další úlohy nechť program vypíše dotaz uživateli, zda pokračovat ve zpracování či nikoli – odpověď uživatele bude znak `'a'` nebo `'n'` (malými nebo velkými písmeny). Program má skončit v případě, že odpověď uživatele je `'n'`. Při načítání počáteční posloupnosti nechť program nejdříve načte počet a poté jednotlivé hodnoty posloupnosti. Poté má program načítat další čísla do zadání záporné hodnoty a tato čísla postupně zařazovat do posloupnosti. Načtením záporného čísla zpracování aktuální úlohy končí, program má vypsat výslednou posloupnost.

   **Ukázka komunikace programu s uživatelem**

```
Pokracovat ve zpracovani (a/n):
a
Zadej pocet prvku pocatecni posloupnosti:
5
Zadej pocatecni vzestupne setridenou posloupnost:
8 12 15 29 64
Zadavej dalsi hodnoty:
5
15
25
28
25
-1
Vysledna posloupnost:
5 8 12 15 25 28 29 64

Pokracovat ve zpracovani (a/n):
y
Pokracovat ve zpracovani (a/n):
Y
Pokracovat ve zpracovani (a/n):
N
```

6. Zapište **program**, který pro každé dva zadané **polynomy vyhodnotí jejich součet a**

**součin**. Vstupní polynomy jsou zadány stupněm a jednotlivými koeficienty.

**Specifikace vstupu**

Program má umožnit při jednom spuštění zpracování libovolného počtu dvojic polynomů. Pro každý polynom bude zadán stupeň a jeho koeficienty. Program se má ukončit v případě, že bude zadán záporný stupeň prvního polynomu.

**Ukázka komunikace programu s uživatelem**

```
Stupen prvního polynomu:
1
Koeficienty prvniho polynomu:
1 5
Stupen druheho polynomu:
1
Koeficienty druheho polynomu:
1 -4
První polynom: x + 5
Druhy polynom: x - 4
Soucet polynomu: 2x + 1
Soucin polynomu: x^2 + x - 20

Stupen prvního polynomu:
0
Koeficienty prvniho polynomu:
12
Stupen druheho polynomu:
3
Koeficienty druheho polynomu:
2 0 0 -5
První polynom: 12
Druhy polynom: 2x^3 - 5
Soucet polynomu: 2x^3 + 7
Soucin      nomu: 24x^3 - 60
Stupen prvního polynomu:
-1
```

7. Zapište **program**, který načte souřadnice vrcholu trojúhelníka. Dále má program načítat body a **určit kolik ze zadaných bodů leží uvnitř, na hranici a kolik vně zadaného trojúhelníka**. Ošetřete testování ostré rovnosti, nerovnosti reálných hodnot zavedením a použitím vhodné konstanty.

**Specifikace vstupu**

Program má umožnit při jednom spuštění zpracování libovolného počtu zadání. Před zadáváním další úlohy nechť program vypíše dotaz uživateli, zda pokračovat ve zpracování či nikoli – odpověď uživatele bude znak `'a'` nebo `'n'` (malými nebo velkými písmeny). Program má skončit v případě, že odpověď uživatele je `'n'`. Při načítání vstupních hodnot nechť program nejdříve načte souřadnice x a y tří vrcholů trojúhelníka. Následně má program načíst počet zpracovávaných bodů, pro které je třeba učit polohu vzhledem k trojúhelníku a poté postupně souřadnice jednotlivých bodů.

**Ukázka komunikace programu s uživatelem**

```
Pokracovat ve zpracovani (a/n):
a
Zadej vrcholy trojuhelnika:
0 0
```

```
2 0
0 2
Zadej pocet testovanych bodu:
4
Zadej souradnice bodu:
1 1
-1 0,45
0,5 0,2
2 2
Uvnitr trojuhelnika lezi 1 bodu
Na hranici trojuhelnika lezi 1 bodu
Vne trojuhelnika lezi 2 bodu

Pokracovat ve zpracovani (a/n):
y
Pokracovat ve zpracovani (a/n):
Y
Pokracovat ve zpracovani (a/n):
N
```

8. Zapište **program**, který bude zadanou **sadu bodů ve 3D třídit dle vzrůstající vzdálenosti jednotlivých bodů od počátku soustavy souřadné**.

**Specifikace vstupu**

Program má umožnit při jednom spuštění zpracování libovolného počtu zadání. Před zadáváním další úlohy nechť program vypíše dotaz uživateli, zda pokračovat ve zpracování či nikoli – odpověď uživatele bude znak `'a'` nebo `'n'` (zadáno malým nebo velkým písmenem). Program má skončit v případě, že odpověď uživatele je `'n'`. Při načítání vstupních hodnot nechť program nejdříve načte počet zpracovávaných bodů, poté jednotlivé body zadané souřadnicemi **x, y, z**.

**Ukázka komunikace programu s uživatelem**

```
Pokracovat ve zpracovani (a/n):
a
Zadej pocet bodu:
4
Zadej souradnice bodu:
10,32 0 5,7
-3 -8 2
15 20 -13
1 0 1
Setridene body
   1,00 0,00 1,00
  -3,00 -8,00 2,00
  10,32 0,00 5,70
  15,00 20,00 -13,00

Pokracovat ve zpracovani (a/n):
y
Pokracovat ve zpracovani (a/n):
Y
Pokracovat ve zpracovani (a/n):
N
```

9. Zapište **program**, který nejprve načte přímku zadanou dvěma body. Poté má program načíst sadu bodů a **setřídit tyto body dle vzrůstající vzdálenosti od přímky**.

**Specifikace vstupu**

Program má umožnit při jednom spuštění zpracování libovolného počtu zadání. Před zadáváním další úlohy nechť program vypíše dotaz uživateli, zda pokračovat ve zpracování či nikoli – odpověď uživatele bude znak `'a'` nebo `'n'` (zadáno malým nebo velkým písmenem). Program má skončit v případě, že odpověď uživatele je `'n'`. Při načítání vstupních hodnot nechť program nejdříve načte body přímky, poté počet zpracovávaných bodů a následně jednotlivé body zadané souřadnicemi *x, y*.

**Ukázka komunikace programu s uživatelem**

```
Pokracovat ve zpracovani (a/n):
A
Zadej body primky:
1 0
-1 0
Zadej pocet bodu:
4
Zadej souradnice bodu:
10,32 0
-3 -8
15 20
1 0
Setridene body:
  10,32 0,00
   1,00 0,00
  -3,00 -8,00
  15,00 20,00

Pokracovat ve zpracovani (a/n):
y
Pokracovat ve zpracovani (a/n):
Y
Pokracovat ve zpracovani (a/n):
N
```

10. Pro každé z celých čísel můžeme vytvořit soupisku cifer čísla. Soupisku čísla vytvoříme tak, že zjistíme počty jednotlivých cifer a soupisku poté sestavíme tak, že cifry s nenulovým počtem zapíšeme ve tvaru počet_cifra_počet_cifra…. Tento zápis bez mezer je opět zápisem celého kladného čísla. Například soupiskou čísla 120651 je 1021121516. Soupiska čísla 31123314 je toto číslo samotné tedy 31123314, obdobně číslo 22 je soupiskou sebe sama. Zapište **program**, který pro každé zadané kladné číslo (menší nebo rovné maximální hodnotě typu `long` jazyka Java) **určí, zda číslo samo je zároveň zápisem své soupisky**.

**Specifikace vstupu**

Program má umožnit při jednom spuštění zpracování libovolného počtu zadání. Program bude postupně načítat čísla a pro každé zadané kladné číslo vypíše jeho soupisku a informaci, zda číslo je či není zároveň zápisem své soupisky. Po načtení záporného nebo nulového čísla nechť program skončí svoji činnost.

**Ukázka komunikace programu s uživatelem**

```
Zadej cislo
22
Cislo 22 ma soupisku 22
Cislo 22 je zapisem sve soupisky

Zadej cislo
103113
```

```
Cislo 103113 ma soupisku 103123
Cislo 103113 neni zapisem sve soupisky

Zadej cislo
31123314
Cislo 31123314 ma soupisku 31123314
Cislo 31123314 neni zapisem sve soupisky

Zadej cislo
-1
```

11. Pro každé z celých čísel můžeme vytvořit soupisku cifer čísla. Soupisku čísla vytvoříme tak, že zjistíme počty jednotlivých cifer a soupisku poté sestavíme tak, že cifry s nenulovým počtem zapíšeme ve tvaru počet_cifra_počet_cifra…. Tento zápis bez mezer je opět zápisem celého kladného čísla. Například soupiskou čísla 120651 je 1021121516. Soupiska čísla 31123314 je toto číslo samotné tedy 31123314, obdobně číslo 22 je soupiskou sebe sama. Zapište **program**, který **ze všech čísel v zadaném rozsahu určí ta čísla, která jsou zároveň zápisem své soupisky**.

**Specifikace vstupu**

Program má umožnit při jednom spuštění zpracování libovolného počtu zadání. Program bude postupně načítat dvojice čísel jako rozsah hodnot, které se mají otestovat. Pro každý zadaný rozsah nechť program vypíše všechna čísla, která jsou svojí vlastní soupiskou. Po načtení záporného nebo nulového čísla namísto první hodnoty rozsahu nechť program skončí svoji činnost.

**Ukázka komunikace programu s uživatelem**

```
Zadej rozsah
10
30
Nalezena cisla
22
Pocet nalezenych cisel 1

Zadej rozsah
-1
```

12. Zapište **program** pro **testování generátoru náhodných čísel**. Generátor náhodných čísel by měl generovat zcela náhodnou posloupnost hodnot, kde aktuálně generovaná hodnota není závislá na hodnotách předchozích. Z náhodně generované posloupnosti hodnot v rozsahu od *a* do *b* vytvořme dvojice čísel $(x_i, y_i)$, kde první z hodnot $x_i$ bude náhodně generované číslo v *i*-tém kroku a druhé z čísel $y_i$ bude náhodně generované číslo v $(i+k)$-tém kroku. Pokud generujeme celkem *n* čísel potom takovýchto dvojic z generované řady lze vytvořit celkem *n-k*. Pro výslednou sadu dvojic můžeme vyhodnotit korelační koeficient. Pro dostatečnou sadu hodnot a pro $k \neq 0$ by měl mít vypočtený korelační koeficient hodnotu blízkou 0. Zapište program, který bude pro zadané hodnoty parametrů *n* a *k* vyhodnocovat korelační koeficient.

Korelační koeficient (Pearsonův korelační koeficient) *r* vypočtěte dle vzorce:

$$r = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}}$$

Poznámka: korelační koeficient odhalí pouze případnou lineární závislost.

**Specifikace vstupu**

Program má umožnit při jednom spuštění zpracování libovolného počtu zadání. Před zadáváním další úlohy nechť program vypíše dotaz uživateli, zda pokračovat ve zpracování či nikoli – odpověď uživatele bude znak `'a'` nebo `'n'` (zadáno malým nebo velkým písmenem). Program má skončit v případě, že odpověď uživatele je `'n'`. Při načítání vstupních hodnot program načte nejdříve meze *a* a *b* intervalu náhodně generovaných čísel a poté dvě celá kladná čísla *n* a *k*.

**Ukázka komunikace programu s uživatelem**

```
Pokracovat ve zpracovani (a/n):
a
Dolni mez intervalu:
100
Horni mez intervalu:
200
Pocet generovanych hodnot:
1000
Delka kroku pro vytvareni dvojic:
2
Vypocteny korelacni koeficient ma hodnotu 0.012

Pokracovat ve zpracovani (a/n):
y
Pokracovat ve zpracovani (a/n):
Y
Pokracovat ve zpracovani (a/n):
N
```

13. Zapište **program** pro **testování generátoru náhodných čísel**. Generujme opakovaně celá čísla v zadaném rozsahu. Pro dostatečně dlouhou řadu vygenerovaných náhodných čísel bychom měli obdržet rovnoměrné pokrytí všech hodnot z příslušného intervalu. Zapište program, který načte počet generovaných hodnot *n* a interval generovaných čísel zadaný jeho krajními hodnotami *a* a *b*. Program má vyhodnotit četnosti jednotlivých hodnot ve vygenerované sadě dat a vypsat:

a) hodnoty a četnosti dle pořadí hodnot,

b) hodnoty a četnosti v pořadí dle dosažených četností,

c) hodnoty s nulovou četností,

d) procentuální zastoupení hodnot s nulovou četností.

**Specifikace vstupu**

Program má umožnit při jednom spuštění zpracování libovolného počtu zadání. Před zadáváním další úlohy nechť program vypíše dotaz uživateli, zda pokračovat ve zpracování či nikoli – odpověď uživatele bude znak `'a'` nebo `'n'` (zadáno malým nebo velkým písmenem). Program má skončit v případě, že odpověď uživatele je `'n'`. Při načítání vstupních hodnot nechť program načte meze intervalu *a* a *b* a počet generovaných čísel *n*.

**Ukázka komunikace programu s uživatelem**

```
Pokracovat ve zpracovani (a/n):
a
Dolni mez intervalu:
100
```

```
Horni mez intervalu:
200
Pocet generovanych hodnot:
20
Vypis dle generovaných hodnot:
...
Vypis dle dosazenych cetnosti
...
Hodnoty s nulovou četností
...
Hodnot s nulovou cetnosti je 82%

Pokracovat ve zpracovani (a/n):
y
Pokracovat ve zpracovani (a/n):
Y
Pokracovat ve zpracovani (a/n):
N
```

14. Je dána pozice *k* dam číslem řádku a číslem sloupce na šachovnici velikosti *n*×*n*. Dvě dámy se navzájem ohrožují, pokud jsou ve stejném řádku, sloupci nebo na stejné diagonále. Zapište program, který pro zadanou úlohu bude zjišťovat všechny dvojice dam, které se navzájem ohrožují.

    **Specifikace vstupu**

    Program má umožnit při jednom spuštění zpracování libovolného počtu zadání. Před zadáváním další úlohy nechť program vypíše dotaz uživateli, zda pokračovat ve zpracování či nikoli – odpověď uživatele bude znak `'a'` nebo `'n'` (zadáno malým nebo velkým písmenem). Program má skončit v případě, že odpověď uživatele je `'n'`. Při načítání vstupních hodnot konkrétní úlohy nechť program načte velikost šachovnice *n*, počet dam *k* a poté pozice jednotlivých dam.

    **Ukázka komunikace programu s uživatelem**

```
Pokracovat ve zpracovani (a/n):
a
Velikost sachovnice:
10
Pocet dam:
4
Pozice jednotlivých dam:
10 5
7 8
7 2
1 6
Kolizni dvojice:
1 2
1 3
2 3

Pokracovat ve zpracovani (a/n):
y
Pokracovat ve zpracovani (a/n):
Y
Pokracovat ve zpracovani (a/n):
N
```

15. Zapište **program**, který **do obdélníkové matice zadané velikosti zapíše čísla** 1..*n***m* po spirále – počínaje „levým horním rohem“, ke středu, ve směru hodinových ručiček.

Hodnoty *n* a *m* představují počet řádků a počet sloupců vytvářené matice.

**Specifikace vstupu**

Program má umožnit při jednom spuštění zpracování libovolného počtu zadání. Program bude postupně načítat dvojice čísel jako velikost vytvářené matice. Pro každé zadání nechť program vypíše výslednou matici. Po načtení záporného nebo nulového čísla namísto prvního rozměru matice nechť program skončí svoji činnost.

**Ukázka komunikace programu s uživatelem**

```
Pocet radku
2
Pocet sloupcu
2
Vysledna matice
1 2
4 3

Pocet radku
4
Pocet sloupcu
3
Vysledna matice
 1 2 3
10 11 4
 9 12 5
 8 7 6

Pocet radku
4
Pocet sloupcu
6
Vysledna matice
  1 2 3 4 5 6
 16 17 18 19 20 7
 15 24 23 22 21 8
 14 13 12 11 10 9

Pocet radku
-1
```

16. Zapište **program**, který **do obdélníkové matice zadané velikosti zapíše čísla** 1..*n***m* po spirále – od středu, proti směru hodinových ručiček. Hodnoty *n* a *m* představují počet řádků a počet sloupců vytvářené matice.

**Specifikace vstupu**

Program má umožnit při jednom spuštění zpracování libovolného počtu zadání. Program bude postupně načítat dvojice čísel jako velikost vytvářené matice. Pro každé zadání nechť program vypíše výslednou matici. Po načtení záporného nebo nulového čísla namísto prvního rozměru matice nechť program skončí svoji činnost.

**Ukázka komunikace programu s uživatelem**

```
Pocet radku
3
Pocet sloupcu
4
Vysledna matice
 12 11 10 9
```

```
  3 2 1 8
  4 5 6 7

Pocet radku
4
Pocet sloupcu
4
Vysledna matice
  16 15 14 13
   5 4 3 12
   6 1 2 11
   7 8 9 10

Pocet radku
-1
```

17. Zapište **program**, který do **obdélníkové matice zadané velikosti *n*×*m* zapíše hodnoty 0 a 1** tak, aby tyto hodnoty v matici tvořily souvislé bloky zadané velikosti *k* uspořádané ve tvaru šachovnice.

    **Specifikace vstupu**

    Program má umožnit při jednom spuštění zpracování libovolného počtu zadání. Program bude postupně načítat dvojice čísel jako velikost vytvářené matice. Pro každé zadání nechť program vypíše výslednou matici. Po načtení záporného nebo nulového čísla namísto prvního rozměru matice nechť program skončí svoji činnost.

    **Ukázka komunikace programu s uživatelem**

```
Pocet radku
6
Pocet sloupcu
6
Velikost bloku
2
Vysledna matice
0 0 1 1 0 0
0 0 1 1 0 0
1 1 0 0 1 1
1 1 0 0 1 1
0 0 1 1 0 0
0 0 1 1 0 0

Pocet radku
7
Pocet sloupcu
7
Velikost bloku
3
Vysledna matice
0 0 0 1 1 1 0
0 0 0 1 1 1 0
0 0 0 1 1 1 0
1 1 1 0 0 0 1
1 1 1 0 0 0 1
1 1 1 0 0 0 1
0 0 0 1 1 1 0

Pocet radku
-1
```

18. Zapište **program**, který bude načítat matice číselných hodnot a v každé zadané matici **uspořádá řádky** tak, aby prvky prvního sloupce tvořily neklesající posloupnost.

**Specifikace vstupu**

Program má umožnit při jednom spuštění zpracování libovolného počtu zadání. Pro každou úlohu program nejdříve načte dvě celá čísla jako počet řádků a počet sloupců zpracovávané matice, poté budou načítány jednotlivé hodnoty matice (po řádcích). Pro každé zadání nechť
program vypíše upravenou matici. Po načtení záporného nebo nulového čísla namísto prvního rozměru matice nechť program skončí svoji činnost.

**Ukázka komunikace programu s uživatelem**

```
Pocet radku
3
Pocet sloupcu
4
Zadej matici
8,2 12,6 123,8 4,1
0 -136,4 12 5
15 19 11 -3
Usporadana matice
    0,00 -136,40 12,00 5,00
    8,20 12,60 123,80 4,10
   15,00 19,00 11,00 -3,00

Pocet radku
-1
```

19. Zapište **program**, který pro každou zadanou matici provede **výměnu dvou řádků a dvou sloupců matice** tak, aby po výměně byl prvek matice s maximální absolutní hodnotou na pozici prvního řádku prvního sloupce (indexy 0, 0).

**Specifikace vstupu**

Program má umožnit při jednom spuštění zpracování libovolného počtu zadání. Pro každou úlohu program nejdříve načte dvě celá čísla jako počet řádků a počet sloupců zpracovávané matice, poté budou načítány jednotlivé hodnoty matice (po řádcích). Pro každé zadání nechť
program vypíše upravenou matici. Po načtení záporného nebo nulového čísla namísto prvního rozměru matice nechť program skončí svoji činnost.

**Ukázka komunikace programu s uživatelem**

```
Pocet radku
3
Pocet sloupcu
4
Zadej matici
26 12,6 123,8 4,1
0 -136,4 12 5
15 19 11 -3

Usporadana matice
   -136,40 0,00 12,00 5,00
     12,60 26,00 123,80 4,10
     19,00 15,00 11,00 -3,00
```

```
Pocet radku
-1
```

20. Zapište **program**, který pro zadanou čtvercovou matici **bude testovat její případnou symetrii**. Program má určovat, zda se jedná o matici symetrickou hodnotami, matici symetrickou strukturou nebo matici nesymetrickou.

    **Zavedení pojmu symetrie matice**

    Nechť čtvercová matice **A** má rozměr *n*, tedy *n* řádků *n* sloupců, prvky matice označíme $a_{i,j}$ kde *i* a *j* jsou hodnoty od 1 do *n* .

    Matice je *symetrická hodnotami* právě tehdy, když $a_{i,j} = a_{j,i}$, pro všechna *i* a *j* od 1 do *n*. Matice je *symetrická strukturou* právě tehdy, když na odpovídajících pozicích (tedy hodnoty prvků $a_{i,j} = a_{j,i}$ ) jsou hodnoty stejného znaménka, tedy buď obě hodnoty jsou nulové, nebo obě kladné, nebo obě záporné.

    Pokud matice není symetrická hodnotami ani symetrická strukturou, potom řekneme, že matice je *nesymetrická*.

    **Specifikace vstupu**

    Program má umožnit při jednom spuštění zpracování libovolného počtu zadání. Pro každou úlohu program celé číslo jako rozměr zpracovávané matice, poté budou načítány jednotlivé hodnoty matice (po řádcích). Pro každé zadání nechť program vypíše informaci,zda se jedná o matici symetrickou hodnotami, symetrickou strukturou nebo nesymetrickou. Po načtení záporného nebo nulového čísla namísto rozměru matice nechť program skončí svoji činnost.

    **Ukázka komunikace programu s uživatelem**

```
Rozmer matice
3
Zadej matici
15 7 16
7 23 -11
16 -11 18
Matice je symetrická hodnotami

Rozmer matice
3
Zadej matici
15 -8 0
-6 3 2
0 13 -18
Matice je symetrická strukturou

Rozmer matice
3
Zadej matici
15 7 16
7 23 -11
16 11 18
Matice je nesymetricka

Rozměr matice
-1
```

21. Zapište program, který načte dvě čtvercové matice stejné velikosti a zjistí, zda druhá matice vznikne z první jednoduchou transformací rotace o 0, 90, 180, 270 stupňů (v pravotočivém směru).

**Specifikace vstupu**

Program má umožnit při jednom spuštění zpracování libovolného počtu zadání. Pro každou úlohu program celé číslo jako rozměr zpracovávaných matic, poté budou načítány jednotlivé matice (po řádcích). Pro každé zadání nechť program vypíše informaci, zda druhá ze zadaných matic vznikne z první některou z uvedených transformací. Program nechť výsledek svého šetření vypisuje jako jednu z následujících krátkých textových zpráv: „Rotace 0“ „Rotace 90“, „Rotace 180“, „Rotace 270“, „Není rotace“ Po načtení záporného nebo nulového čísla namísto rozměru matice nechť program skončí svoji činnost.

**Poznámka**: pro dvě zadané matice může být nalezeno i více transformací.

**Ukázka komunikace programu s uživatelem**

```
Rozmer matic
3
Prvni matice
1 2 3
4 5 6
7 8 9
Druha matice
7 4 1
8 5 2
9 6 3
Rotace 270

Rozmer matic
3
Prvni matice
1 2 3
4 5 6
7 8 9
Druha matice
3 6 9
2 5 8
1 4 7
Rotace 90

Rozmer matic
3
Prvni matice
1 2 3
4 5 6
7 8 9
Druha matice
1 4 7
2 5 8
3 6 9
Není rotace

Rozměr matice
-1
```

22. Zapište program, který načte dvě čtvercové matice stejné velikosti a zjistí, zda druhá matice vznikne z druhé jednoduchou transformací zrcadlení podle hlavní nebo vedlejší diagonály nebo „prostředního“ řádku nebo sloupce (lépe horizontální, vertikální osy).

**Specifikace vstupu**

Program má umožnit při jednom spuštění zpracování libovolného počtu zadání. Pro každou úlohu program celé číslo jako rozměr zpracovávaných matic, poté budou načítány jednotlivé matice (po řádcích). Pro každé zadání nechť program vypíše informaci, zda druhá ze zadaných matic vznikne z první některou z uvedených transformací. Program nechť výsledek svého šetření vypisuje jako jednu z následujících krátkých textových zpráv: „Zrcadlení dle hlavní diagonály“, „Zrcadlení dle vedlejší diagonály“, „Zrcadlení dle vertikální osy“, „Zrcadlení dle horizontální osy“, „Nenalezena transformace“ Po načtení záporného nebo nulového čísla namísto rozměru matice nechť program skončí svoji činnost.

**Poznámka**: pro dvě zadané matice může být nalezeno i více transformací.

**Ukázka komunikace programu s uživatelem**

```
Rozmer matic
3
Prvni matice
1 2 3
4 5 6
7 8 9
Druha matice
7 8 9
4 5 6
1 2 3
Zrcadlení dle horizontální osy

Rozmer matic
3
Prvni matice
1 2 3
4 5 6
7 8 9
Druha matice
3 6 9
2 5 8
1 4 7
Nenalezena transformace

Rozmer matic
3
Prvni matice
1 2 3
4 5 6
7 8 9
Druha matice
1 4 7
2 5 8
3 6 9
Zrcadlení dle hlavni diagonaly

Rozměr matice
-1
```

23. Zapište **program**, který načte matici a bude provádět zadané **transformace zadané matice o 0, -90 a 90 stupňů**. Transformace budou zadávány hodnotami 0, -1, 1.

    **Specifikace vstupu**

    Program má umožnit při jednom spuštění zpracování libovolného počtu zadání. Pro každou úlohu program nejdříve načte dvě celá čísla jako počet řádků a počet sloupců

matice a poté vlastní matici. Dále bude program postupně načítat čísla 0, 1 nebo -1 a pro každé zadané číslo provede příslušnou transformaci. Zadáním jiného čísla končí zpracování aktuální úlohy. Po načtení záporného nebo nulového čísla namísto počtu řádků matice nechť program skončí svoji činnost.

**Ukázka komunikace programu s uživatelem**

```
Pocet radku
3
Pocet sloupcu
3
Matice
1 2 3
4 5 6
7 8 9
Transformace
1
Rotovaná matice
3 6 9
2 5 8
1 4 7
Transformace
1
Rotovaná matice
9 8 7
6 5 4
3 2 1
Transformace
0
Rotovaná matice
9 8 7
6 5 4
3 2 1
Transformace
2

Pocet radku
-1
```

24. Zapište **program**, který bude provádět redukci čtvercové matice. V načtené matici má program vyhledat nenulový prvek pro který platí, že je jediným nenulovým prvkem na řádku a ve sloupci. Pokud takový prvek v matici neexistuje, potom má program vypsat informaci, že matici nelze dále upravit. Pokud je prvek nalezen, potom má program odstranit řádek a sloupec, ve kterém se nalezený prvek nachází a opakovaně přejít k vyhledání dalšího prvku v již upravené matici, který splňuje uvedenou podmínku.

**Specifikace vstupu**

Program má umožnit při jednom spuštění zpracování libovolného počtu zadání. Pro každou úlohu program celé číslo jako rozměr zpracovávané matice, poté budou načítány jednotlivé hodnoty matice (po řádcích). Pro každé zadání nechť program vypíše redukovanou matici. Po načtení záporného nebo nulového čísla namísto rozměru matice nechť program skončí svoji činnost.

**Ukázka komunikace programu s uživatelem**

```
Rozmer matice
3
Zadej matici
15 0 16
```

```
0 23 0
16 0 18
Redukovana matice (2 x 2)
15 16
16 18

Rozmer matice
3
Zadej matici
15 0 16
11 23 0
16 0 18
Redukovana matice (3 x 3)
15 0 16
11 23 0
16 0 18

Rozměr matice
-1
```

25. Zapište **program**, který bude testovat, zda zadaná čtvercová matice tvoří *latinský čtverec* či nikoli. *Latinský čtverec* je čtvercové schéma $n \times n$ čísel 1, 2, … $n$ takové, že každý řádek a každý sloupec obsahuje všechna čísla od 1 do $n$. To znamená, že všechna čísla v libovolném řádku jsou různá; totéž platí o sloupcích.

    **Specifikace vstupu**

    Program má umožnit při jednom spuštění zpracování libovolného počtu zadání. Pro každou úlohu program načte celé číslo jako rozměr matice poté načte vlastní matici (po řádcích). Pro každé zadání nechť program vypíše výsledek svého šetření jako jednu z následujících krátkých textových zpráv: „Matice tvori latinsky ctverec“, „Nejedna se o latinsky ctverec“. Po načtení záporného nebo nulového čísla namísto rozměru matice nechť program skončí svoji činnost.

    **Ukázka komunikace programu s uživatelem**

```
Rozmer matic
3
Matice
1 2 3
3 1 2
2 3 1
Matice tvori latinsky ctverec

Rozmer matic
4
Matice
1 2 3 4
4 3 2 1
3 4 1 2
2 1 4 3
Matice tvori latinsky ctverec

Rozmer matic
3
Matice
1 2 3
4 5 6
7 8 9
Nejedna se o latinsky ctverec
```

```
Rozměr matice
-1
```

26. Zapište **program**, který bude testovat, zda zadaná čtvercová matice celých čísel tvoří *magický čtverec* či nikoli. Řekneme, že čtvercová matice celých čísel velikosti *n* (*n* řádků, *n* sloupců, matice obsahuje celkem $n^2$ prvků) tvoří *magický čtverec* právě, když (a) součty ve všech řádcích, ve všech sloupcích a v obou diagonálách jsou stejné a zároveň (b) matice obsahuje všechny hodnoty 1, 2, 3, … $n^2$.

**Specifikace vstupu**

Program má u možnit při jednom spuštění zpracování libovolného počtu zadání. Pro každou úlohu nechť program načte celé číslo jako rozměr matice poté načte vlastní matici (po řádcích). Pro každé zadání nechť program vypíše výsledek svého šetření jako jednu z následujících krátkých textových zpráv: „Matice tvori magicky ctverec“, „Nejedna se o magicky ctverec“. Po načtení záporného nebo nulového čísla namísto rozměru matice nechť program skončí svoji činnost.

**Ukázka komunikace programu s uživatelem**

```
Rozmer matic
3
Prvni matice
4 9 2
3 5 7
8 1 6
Matice tvori magicky ctverec

Rozmer matic
4
Prvni matice
 7 12 1 14
 2 13 8 11
16 3 10 5
 9 6 15 4
Matice tvori magicky ctverec

Rozmer matic
3
Prvni matice
1 2 3
4 5 6
7 8 9
Nejedna se o magicky ctverec

Rozměr matice
-1
```

27. Je zadáno *n* vektorů délky *k*. Zapište **program**, který pro každou takovouto sadu vektorů **nalezne a vypíše dva vektory s maximálním skalárním součinem**.

Pro dva vektory $\mathbf{u} = (u_1, u_2, \ldots, u_n)$ a $\mathbf{v} = (v_1, v_2, \ldots, v_n)$ vypočteme *skalární součin* dle vztahu

$$\mathbf{u} \cdot \mathbf{v} = u_1 \cdot v_1 + u_2 \cdot v_2 + \ldots + u_n \cdot v_n = \sum_{i=1}^{n} u_i \cdot v_i$$

**Specifikace vstupu**

Program má umožnit při jednom spuštění zpracování libovolného počtu zadání. Pro

každou úlohu nechť program nejdříve načte celá čísla *n* a *k* (představující počet vektorů a počet složek každého z vektorů). Poté program načte jednotlivé vektory a vypíše výsledek svého šetření. Po načtení záporného nebo nulového *n* nechť program skončí svoji činnost.

**Ukázka komunikace programu s uživatelem**

```
Pocet vektoru
5
Delka vektoru
4
Zadej vektory
0 1 1,2 4
0 0 0 12
1 3 0 0
8 0 0 0
1 1 1 1
Vektory s maximalnim skalarnim soucinem
 (0,00 1,00 1,20 4,00)
 (0,00 0,00 0,00 12,00)
Skalarni soucin techto vektoru 48,00

Pocet vektoru
-1
```

28. Zapište **program**, který bude **testovat, zda zadaný systém *n* vektorů o *n* složkách je ortonormální**.

Systém vektorů je *ortonormální* právě tehdy, když všechny vzájemné skalární součiny dvojice různých vektorů jsou rovny 0 a zároveň každý z vektorů má délku 1.

Uvažujme *vektor* **u** v *n*-rozměrném prostoru: $\mathbf{u} = (u_1, u_2, \ldots, u_n)$.
*Délku vektoru* (normu vektoru) **u** označíme ||**u**|| a vypočteme ji dle vztahu

$$\|\mathbf{u}\| = \sqrt{u_1^2 + u_2^2 + \ldots + u_n^2} = \sqrt{\sum_{i=1}^{n} u_i^2}$$

.

Pro dva vektory $\mathbf{u} = (u_1, u_2, \ldots, u_n)$ a $\mathbf{v} = (v_1, v_2, \ldots, v_n)$ vypočteme *skalární součin* dle vztahu

$$\mathbf{u} \cdot \mathbf{v} = u_1 \cdot v_1 + u_2 \cdot v_2 + \ldots + u_n \cdot v_n = \sum_{i=1}^{n} u_i \cdot v_i$$

**Poznámka:** Pozor na testování ostré rovnosti reálných čísel.

**Specifikace vstupu**

Program má umožnit při jednom spuštění zpracování libovolného počtu zadání. Pro každou úlohu nechť program nejdříve načte celé číslo *n* (představující počet vektorů i počet složek každého z vektorů). Poté program načte jednotlivé vektory a vypíše výsledek svého šetření v podobě krátké zprávy tvaru „System vektoru je ortonormalni“ nebo „System vektoru není ortonormalni“. Po načtení záporného nebo nulového *n* nechť program skončí svoji činnost.

**Ukázka komunikace programu s uživatelem**

```
Pocet vektoru
4
Zadej vektory
1 0 0 0
```

```
0 -1 0 0
0 0 0 1
0 0 1 0
System vektoru je ortonormalni

Pocet vektoru
3
Zadej vektory
2 3,1 0
0 -1 0
0 0 1
System vektoru neni ortonormalni

Pocet vektoru
-1
```

29. Zapište **program**, který bude **testovat, zda zadaný systém *n* vektorů o *n* složkách je ortogonální**. Pokud ano nechť program dále **provede převod vektorů na normovaný tvar.**

*Délku vektoru* (normu vektoru) $\mathbf{u} = (u_1, u_2, \ldots, u_n)$ označíme ||**u**|| a vypočteme ji dle vztahu $\|\mathbf{u}\| = \sqrt{u_1^2 + u_2^2 + \ldots + u_n^2} = \sqrt{\sum_{i=1}^{n} u_i^2}$.

*Normovaným vektorem* $\mathbf{u}_n$ k vektoru **u** nazveme vektor, který má délku 1 a jehož směr je totožný se směrem původního vektoru. Normovaný tvar získáme vydělení každé složky původního vektoru jeho normou, tedy

$$\mathbf{u}_n = \left(\frac{u_1}{\|\mathbf{u}\|}, \frac{u_2}{\|\mathbf{u}\|}, \ldots, \frac{u_n}{\|\mathbf{u}\|}\right).$$

Pro dva vektory $\mathbf{u} = (u_1, u_2, \ldots, u_n)$ a $\mathbf{v} = (v_1, v_2, \ldots, v_n)$ vypočteme *skalární součin* dle vztahu $\mathbf{u} \cdot \mathbf{v} = u_1 \cdot v_1 + u_2 \cdot v_2 + \ldots + u_n \cdot v_n = \sum_{i=1}^{n} u_i \cdot v_i$.

Systém *n* vektorů o *n* složkách nazveme *ortogonálním systémem* právě tehdy, když všechny vzájemné skalární součiny (dvou navzájem různých! vektorů) jsou rovny 0 a žádný z vektorů nemá nulovou délku.

**Poznámka:** Pozor na testování ostré rovnosti reálných čísel.

**Specifikace vstupu**

Program má umožnit při jednom spuštění zpracování libovolného počtu zadání. Pro každou úlohu nechť program nejdříve načte celé číslo *n* (představující počet vektorů i počet složek každého z vektorů). Poté program načte jednotlivé vektory a vypíše výsledek testu ortogonality v podobě krátké zprávy tvaru „System je ortogonalni“ nebo „System není ortogonální“. Po načtení záporného nebo nulového *n* nechť program skončí svoji činnost.

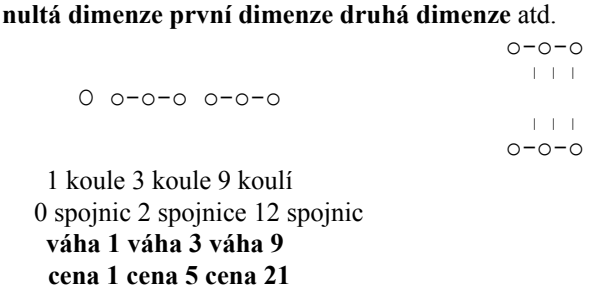
**Ukázka komunikace programu s uživatelem**

```
Pocet vektoru
4
Zadej vektory
```

```
2 0 0 0
0 5 0 0
0 0 0 4
0 0 3,6 0
System je ortogonalni
Normalizovany system
1 0 0 0
0 1 0 0
0 0 0 1
0 0 1 0

Pocet vektoru
3
Zadej vektory
2 3,1 0
0 -1 0
0 0 1
System neni ortogonalni
Pocet
vektoru
-1
```

30. **Program „Lodě“**. Je třeba vyřešit problém,jak co nejvýhodněji využívat vesmírné nákladní lodě. Lodě mají přepravovat předměty ve tvaru d-dimenzionální mřížky, jejichž rozměr v každé dimenzi je 3. V uzlech mřížky jsou koule jednotkové váhy. Mezi uzly vedou spojnice, jejichž hmotnost je vzhledem k hmotnosti koulí zanedbatelná. Váha každého předmětu je tedy rovna počtu jeho uzlů. Cena každého předmětu je rovna součtu počtu uzlů a počtu spojnic. Každá loď má určitou nosnost a my chceme přepravu organizovat tak, aby při každé jízdě měl náklad maximální cenu.




Vaším úkolem je napsat **program**, který **pro zadanou nosnost určí, kolik předmětů a jaké dimenze je třeba naložit**, aby cena těchto předmětů byla maximální. Předpokládejte, že máte k dispozici neomezený počet kusů od každého předmětu. K dispozici jsou předměty do dimenze 10. Zadaná nosnost nebude větší než 2147483647.

**Specifikace vstupu**

Program má umožnit při jednom spuštění zpracování libovolného počtu zadání – pro každé celé kladné cele číslo představující nosnost lodě vypsat počty předmětů jednotlivých dimenzí. Po načtení záporného nebo nulového čísla namísto nosnosti lodě nechť program skončí svoji činnost.

**Ukázka komunikace programu s uživatelem**

```
Nosnost lode
100
Pocty predmetu jednotlivych dimenzi
1 0 2 0 1
```

```
Nosnost lode
175
Pocty predmetu jednotlivych dimenzi
2 0 1 1 1

Nosnost lode
9841
Pocty predmetu jednotlivych dimenzi
1 1 1 1 1 1 1 1 1

Nosnost lode
-1
```