Investice do pozemků

Termín odevzdání:	**03.12.2017 23:59:59**
Hodnocení:	**6.6000**
Max. hodnocení:	**5.0000** (bez bonusů)
Odevzdaná řešení:	7 / 20 Volné pokusy + 10 Penalizované pokusy (-10 % penalizace za každé odevzdání)
Nápovědy:	2 / 2 Volné nápovědy + 2 Penalizované nápovědy (-10 % penalizace za každou nápovědu)

Úkolem je vytvořit program, který bude vyhledávat optimální investice do pozemků.

Na vstupu programu je zadání cen parcel. Předpokládáme, že parcely leží v pravoúhlém rastru, kde známe počet řádek a sloupců. Parcela má jednotkovou výměru, pro každou parcelu známe její cenu (celé kladné číslo). Po zadání cen pozemků následuje seznam dotazů. Chceme investovat zadaný objem peněz a hledáme parcelu/parcely, které mají v součtu cenu nižší nebo rovnou zadanému objemu peněz. Investice má ale velmi přísná pravidla:

- jsme omezeni tím, že můžeme nakupovat pouze sousední parcely,
- zakoupené parcely navíc musí tvořit obdélník či čtverec v rastru,
- chceme získat pozemek o co největší celkové výměře při daném objemu investice.

Program dokáže zpracovávat dotazy dvou typů: buď pouze zobrazí počet různých možností, jak investovat zadanou částku (dotaz `count`), nebo navíc vypíše i seznam alokací parcel, které dotazu vyhovují (dotaz `list`).

Vstupem programu je:

- velikost rastru (šířka, výška), velikost je omezena na 1 až 2000 v každém směru,
- ceny jednotlivých parcel, ceny jsou zadané po řádcích,
- seznam dotazů.

Dotaz je buď typu `count investment` nebo `list investment`, kde `investment` je investovaná částka.

Výstupem programu je vyřešení dotazů:

- na dotaz typu `count investment` je odpovědí maximální výměra pozemku, který lze za investovanou částku `investment` koupit a počet různých způsobů, kterými toho lze dosáhnout,
- na dotaz typu `list investment` je odpověď stejná jako na předchozí dotaz, program navíc vypíše ještě seznam možných alokací parcel (lze je pořídit za cenu nejvýše `investment` a mají celkovou výměru stejnou jako nalezené maximum). Seznam alokací parcel má podobu:

  ```
  cost: w x h @ (x,y)
  ```

  `cost` ≤ `investment` je cena dané alokace parcel, `w,h` je šířka a výška alokovaných parcel a `x,y` je souřadnice levého horního rohu alokace parcel.
- Pokud je investovaná částka příliš malá, nepůjde koupit žádná parcela. Odpověď programu je pak speciální, viz ukázka.

Pokud je vstup neplatný, program to musí detekovat a zobrazit chybové hlášení. Chybové hlášení zobrazujte na standardní výstup (ne na chybový výstup). Za chybu považujte:

- rozměr rastru je nečíselný, nulový, záporný nebo překračuje limit 2000,
- zadaná cena parcel není číslo, je nulová nebo záporná,
- dotaz není typu `count` ani `list`,
- investovaná částka v dotazu chybí, není správně zadaná, je nulová nebo záporná.

Před implementací programu si rozmyslete, jakým způsobem budete reprezentovat ceny pozemků a jak budete v zadaném rastru vyhledávat. Velikost rastru je omezená na max. 2000 prvků v každém směru. Pro řešení tedy postačuje staticky alokovaná paměť.

Vyhledávání v cenách pozemků může trvat velmi dlouho. Naivní řešení má složitost n^6, vylepšováním algoritmu se dá složitost výrazně snížit. Časové limity testovacího prostředí jsou nastavené tak, aby rozumná implementace naivního algoritmu prošla všemi testy mimo testů bonusových.

Ukázka práce programu:

```
Velikost mapy:
5 6
Cenova mapa:
 13   9  16  14   3
 11   7   5  14   9
```

```
  2   5   9   9   4
 11  13   3   8  16
 19  15   9   4   2
  3   7   8  11  27
```
Dotazy:
count 20
Max. rozloha: 3 (x 6)
count 50
Max. rozloha: 6 (x 15)
count 190
Max. rozloha: 20 (x 6)
list 1
Nenalezeno.
list 2
Max. rozloha: 1 (x 2)
2: 1 x 1 @ (0,2)
2: 1 x 1 @ (4,4)
list 3
Max. rozloha: 1 (x 5)
2: 1 x 1 @ (0,2)
3: 1 x 1 @ (0,5)
3: 1 x 1 @ (2,3)
3: 1 x 1 @ (4,0)
2: 1 x 1 @ (4,4)
list 10
Max. rozloha: 2 (x 3)
7: 2 x 1 @ (0,2)
10: 2 x 1 @ (0,5)
6: 2 x 1 @ (3,4)
list 60
Max. rozloha: 8 (x 1)
60: 4 x 2 @ (0,2)
list 180
Max. rozloha: 20 (x 3)
173: 4 x 5 @ (0,1)
175: 5 x 4 @ (0,1)
174: 4 x 5 @ (1,0)
list 175
Max. rozloha: 20 (x 3)
173: 4 x 5 @ (0,1)
175: 5 x 4 @ (0,1)
174: 4 x 5 @ (1,0)
list 173
Max. rozloha: 20 (x 1)
173: 4 x 5 @ (0,1)
list 172
Max. rozloha: 18 (x 3)
165: 3 x 6 @ (0,0)
166: 3 x 6 @ (1,0)
171: 3 x 6 @ (2,0)
count 1000
Max. rozloha: 30 (x 1)

Velikost mapy:
12 1
Cenova mapa:
1 2 3
4 1 2 3 4 1 2
3 4
Dotazy:
count 10
Max. rozloha: 4 (x 9)
list 9
Max. rozloha: 3 (x 10)
6: 3 x 1 @ (0,0)
9: 3 x 1 @ (1,0)
8: 3 x 1 @ (2,0)
7: 3 x 1 @ (3,0)
6: 3 x 1 @ (4,0)
9: 3 x 1 @ (5,0)
8: 3 x 1 @ (6,0)
7: 3 x 1 @ (7,0)

```
6: 3 x 1 @ (8,0)
9: 3 x 1 @ (9,0)
count 8
Max. rozloha: 3 (x 7)
list 12
Max. rozloha: 5 (x 4)
11: 5 x 1 @ (0,0)
12: 5 x 1 @ (1,0)
11: 5 x 1 @ (4,0)
12: 5 x 1 @ (5,0)
list 0
Nespravny vstup.
```

```
Velikost mapy:
2 2
Cenova mapa:
1 2 3 test
Nespravny vstup.
```

Poznámky:

- Ukázkové běhy zachycují očekávané výpisy Vašeho programu (tučné písmo) a vstupy zadané uživatelem (základní písmo). Zvýraznění tučným písmem je použité pouze zde na stránce zadání, aby byl výpis lépe čitelný. Váš program má za úkol pouze zobrazit text bez zvýrazňování (bez HTML markupu).
- Znak odřádkování (\n) je i za poslední řádkou výstupu (i za případným chybovým hlášením).
- Pro reprezentaci cen parcel postačuje datový typ `int`.
- Souřadnice (0,0) odpovídá levému hornímu rohu mapy, souřadnice rostou směrem vpravo a směrem dolů..
- Dynamická alokace není v této úloze potřeba. Maximální velikost rastru je omezená, paměťové limity postačují pro statickou alokaci. Je ale možné, že se některé reprezentace cenové mapy nevejdou do lokálních proměnných. V takovém případě může být řešením cenovou mapu alokovat v datovém segmentu (hint: klíčové slovo `static`).
- Odřádkování v cenové mapě na vstupu může, ale nemusí respektovat velikost rastru. Program se při načítání vstupu nemusí odřádkováním zabývat, pro zpracování mu stačí dříve zadaná velikost rastru.
- Pro velké velikosti rastru v bonusových testech je nalezeno velké množství vyhovujících alokací. Samotný výpis těchto alokací je dost pomalý. Proto se v bonusových testech kontrolují převážně dotazy typu `count`.
- Pořadí vyhovujících alokací v dotazech typu `list` není určené. Vaše implementace může nalezené alokace vypisovat v libovolném pořadí, testovací prostředí si před porovnáním pořadí ve výpisu upraví. Tedy například pro zadání:

  ```
  Velikost mapy:
  6 4
  Cenova mapa:
  1 2 1 2 1 2
  2 1 2 1 2 1
  1 2 1 2 1 2
  2 1 2 1 2 1
  Dotazy:
  list 3
  ```

  jsou přípustné odpovědi:

  ```
  Max. rozloha: 2 (x 38)
  3: 2 x 1 @ (0,0)
  3: 2 x 1 @ (1,0)
  3: 2 x 1 @ (2,0)
  3: 2 x 1 @ (3,0)
  3: 2 x 1 @ (4,0)
  3: 1 x 2 @ (0,0)
  3: 1 x 2 @ (1,0)
  3: 1 x 2 @ (2,0)
  3: 1 x 2 @ (3,0)
  3: 1 x 2 @ (4,0)
  3: 1 x 2 @ (5,0)
  3: 2 x 1 @ (0,1)
  3: 2 x 1 @ (1,1)
  3: 2 x 1 @ (2,1)
  3: 2 x 1 @ (3,1)
  3: 2 x 1 @ (4,1)
  3: 1 x 2 @ (0,1)
  3: 1 x 2 @ (1,1)
  3: 1 x 2 @ (2,1)
  3: 1 x 2 @ (3,1)
  3: 1 x 2 @ (4,1)
  ```

```
3: 1 x 2 @ (5,1)
3: 2 x 1 @ (0,2)
3: 2 x 1 @ (1,2)
3: 2 x 1 @ (2,2)
3: 2 x 1 @ (3,2)
3: 2 x 1 @ (4,2)
3: 1 x 2 @ (0,2)
3: 1 x 2 @ (1,2)
3: 1 x 2 @ (2,2)
3: 1 x 2 @ (3,2)
3: 1 x 2 @ (4,2)
3: 1 x 2 @ (5,2)
3: 2 x 1 @ (0,3)
3: 2 x 1 @ (1,3)
3: 2 x 1 @ (2,3)
3: 2 x 1 @ (3,3)
3: 2 x 1 @ (4,3)
```

nebo:

```
Max. rozloha: 2 (x 38)
3: 1 x 2 @ (0,0)
3: 1 x 2 @ (1,0)
3: 1 x 2 @ (2,0)
3: 1 x 2 @ (3,0)
3: 1 x 2 @ (4,0)
3: 1 x 2 @ (5,0)
3: 2 x 1 @ (0,0)
3: 2 x 1 @ (1,0)
3: 2 x 1 @ (2,0)
3: 2 x 1 @ (3,0)
3: 2 x 1 @ (4,0)
3: 1 x 2 @ (0,1)
3: 1 x 2 @ (1,1)
3: 1 x 2 @ (2,1)
3: 1 x 2 @ (3,1)
3: 1 x 2 @ (4,1)
3: 1 x 2 @ (5,1)
3: 2 x 1 @ (0,1)
3: 2 x 1 @ (1,1)
3: 2 x 1 @ (2,1)
3: 2 x 1 @ (3,1)
3: 2 x 1 @ (4,1)
3: 1 x 2 @ (0,2)
3: 1 x 2 @ (1,2)
3: 1 x 2 @ (2,2)
3: 1 x 2 @ (3,2)
3: 1 x 2 @ (4,2)
3: 1 x 2 @ (5,2)
3: 2 x 1 @ (0,2)
3: 2 x 1 @ (1,2)
3: 2 x 1 @ (2,2)
3: 2 x 1 @ (3,2)
3: 2 x 1 @ (4,2)
3: 2 x 1 @ (0,3)
3: 2 x 1 @ (1,3)
3: 2 x 1 @ (2,3)
3: 2 x 1 @ (3,3)
3: 2 x 1 @ (4,3)
```

nebo libovolná další ze zbývajících 523022617466601111760007224100074291199999998 permutací.

- Slovní popis struktury platných vstupních dat není zcela exaktní. Proto připojujeme i formální popis vstupního jazyka v EBNF:

```
input      ::= { whiteSpace } gridSize { whiteSpace } priceMap { whiteSpace } queryList
whiteSpace ::= ' ' | '\t' | '\n' | '\r'
gridSize   ::= integer { whiteSpace } integer
priceMap   ::= integer { { whiteSpace } integer }
queryList  ::= { query { whiteSpace } }
query      ::= ( 'list' | 'count' ) { whiteSpace } integer
integer    ::= [ '+' ] digit { digit }
digit      ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
```