File, List, dan *Function*

Tutorial 5 [G, F] - Dasar-Dasar Pemrograman 1 Gasal 2019/2020

Selamat datang di Tutorial 5 DDP1! Sesi tutorial ini akan memperkenalkan kalian pada konsep *file*. Anda juga akan diberikan latihan untuk melakukan *debugging* pada kode. Konsep yang sudah diajarkan sebelumnya tentunya tidak akan dilupakan pada tutorial kali ini, dan juga tutorial seterusnya.

Mohon kumpulkan semua file jawaban Anda dalam bentuk zip dengan format Lab5_Nama_NPM_KodeAsdos.zip. Contoh: Lab5_WindiChandra_1606862721_YE.zip

Anda diperbolehkan untuk tidak menambahkan *file* input sebagai *submission* latihan 14.

Judul	File Submission	Bobot Nilai
Latihan 14: Submatriks	latihan14.py	90
Latihan 15: Bantuin Dong, Kak!	latihan15.py	10
Challenge 4: Minesweeper	challenge4.py	0

File

Selain membaca *input* dari masukan pengguna, program juga dapat membaca *input* dari sebuah *file*. Untuk membaca isi dari sebuah *file*, perintah yang umum digunakan adalah sebagai berikut:

```
f = open('nama-file', 'r')
```

Perintah **open('nama-file', 'r')** akan membuka *file* dengan **'nama-file'** untuk dibaca program, lalu kita perlu menyimpannya pada suatu variabel agar dapat kita gunakan. Pada contoh di atas, variabel yang digunakan adalah **f**. Selanjutnya untuk membaca seluruh isi dari *file*, kita dapat menggunakan perintah **f.read()**. Cobalah membuat sebuah *file* bernama **input.txt** **pada folder yang sama** dengan kode yang anda buat kemudian jalankan kode di bawah ini. Ubahlah isi **input.txt** dengan apa saja dan lihat apa yang dikeluarkan program anda.

```
f = open('input.txt', 'r')

print(f.read())

f.close() # perintah ini untuk menutup file yang sedang dibuka, jalankan perintah ini setiap kali anda selesai membaca file.
```

Proses per baris

Umumnya dalam membaca *file*, kita tidak hanya memproses isi *file* itu secara utuh tetapi kita mungkin memrosesnya per baris, per kata, dan sebagainya. Terdapat beberapa cara untuk memroses *file* per baris.

1. Dengan **f.read()** dan ***split***

   Perintah **f.read()** mengembalikan sebuah *string* yang dapat kita simpan di suatu variabel, ataupun kita **split** dengan pemisah **\n** (newline) untuk memisahkan setiap barisnya.

   ```
   f = open('input.txt', 'r')

   for baris in f.read().split('\n'):
       # kode anda disini

   f.close()
   ```

2. Dengan **for**

   Menariknya, kita dapat langsung melakukan *loop* **for** pada variabel **f** dan kita akan melakukan *loop* per baris. (**Catatan:** dengan cara ini, karakter **\n** (newline) di akhir *string* tidak akan dihapus. Anda dapat melakukan *slicing* **baris[:-1]** jika anda tidak ingin memproses karakter **\n**)

```
f = open('input.txt', 'r')

for baris in f:
    # kode anda disini

f.close()
```

3. Dengan **f.readlines()**

   f.readlines() akan memberikan sebuah *list* berisi *string* setiap barisan, dengan kata lain kita dapat melakukan *looping* pada *list* ini. (**Catatan:** dengan cara ini, karakter **\n** (newline) di akhir *string* tidak akan dihapus. Anda dapat melakukan *slicing* **baris[:-1]** jika anda tidak ingin memproses karakter **\n** atau menggunakan method **rstrip()** untuk menghapus *whitespace* di kanan *string*)

```
f = open('input.txt', 'r')

for baris in f.readlines():
    # kode anda disini

f.close()
```

Selain cara-cara di atas, terdapat juga beberapa cara lain yang tidak dibahas. Cobalah cara-cara di atas dan soal tutorial ini agar lebih paham. Anda juga disarankan untuk mencetak (**print**) hasil dari perintah-perintah yang baru anda ketahui ataupun kurang mengerti. Metode ini juga dapat anda terapkan pada saat mengerjakan soal untuk menemukan kesalahan program anda. Pastikan setiap barisan program anda bekerja dengan benar dengan mengamati keluaran dari perintah **print**.

Method / Function

Setelah mengerjakan tutorial-tutorial sebelumnya, pernahkah kalian menemukan beberapa kode mirip yang harus kalian panggil berulang-ulang? Perhatikan contoh berikut:

```
print('Saya makan 3 apel')
print('Saya makan 50 apel')
print('Saya makan 10 apel')
```

Perhatikan bahwa setiap baris kode diatas dapat kita ubah menjadi **print('Saya makan', x, 'apel')** sehingga kita hanya perlu menentukan nilai **x**. Kita dapat mengubah kode di atas menjadi seperti berikut:

```
x = 3
print('Saya makan', x, 'apel')
x = 50
print('Saya makan', x, 'apel')
x = 10
print('Saya makan', x, 'apel')
```

Perhatikan bahwa barisan perintah **print** yang kita miliki menjadi sama, sekarang kita dapat mendefinisikan sebuah *method* untuk perintah **print** kita tersebut. Contoh:

```
def makan(x):
    print('Saya makan', x, 'apel')
```

Method yang kita definisikan akan menerima sebuah parameter bernama **x**, kemudian menjalankan perintah **print** seperti kode kita sebelumnya. Selanjutnya kode sebelumnya dapat kita modifikasi dengan memanggil *method* yang telah kita buat:

```
makan(3)
makan(50)
makan(10)
```

Selain itu, kita juga dapat mengembalikan sebuah nilai melalui *method*: Mari kita coba mengubah *method* **makan** kita untuk mengembalikan *string* "Saya makan x apel" daripada langsung dicetak:

```
def makan(x):
    return 'Saya makan ' + str(x) + ' apel'
```

return digunakan untuk mengembalikan nilai dari suatu *method*, sehingga fungsi makan kita akan mengembalikan nilai *string*. Selanjutnya, kode sebelumnya dapat kita modifikasi lagi dengan *method* **makan** yang baru.

```
print(makan(3))
print(makan(50))
print(makan(10))
```

Apabila kalian merasa familiar dengan penggunaan *method*, penyebabnya adalah karena kita telah menggunakan *method* sebelumnya, seperti *method* pada *turtle* seperti **turtle.forward()**, **turtle.left()**, kemudian *method* lainnya seperti **max()** ataupun **sort**(). Tentunya **print()** merupakan sebuah *method*.

Latihan #14: Submatriks (latihan14.py, skor: +90)

Konsep penting: file, string, list

Masih ingatkah Anda dengan konsep matriks? Matriks bisa dikatakan sebagai kumpulan bilangan yang diletakkan setiap beberapa baris (atas ke bawah) dan setiap beberapa kolom (kiri ke kanan). Berikut adalah contoh matriks 4x3 yang mempunyai 4 baris dan 3 kolom.

$$\begin{pmatrix} 5 & 3 & 2 \\ 8 & 1 & 3 \\ 7 & 3 & 99 \\ 100 & 11 & 7 \end{pmatrix}$$

Penomoran kali ini menggunakan ***1-based indexing***, yang artinya adalah angka paling kiri atas adalah baris pertama dan kolom pertama. Dalam contoh matriks di atas, nilai pada baris pertama dan kolom pertama adalah **5**.

Kita akan mempelajari tentang **submatriks**. Submatriks dari suatu matriks adalah 'potongan' matriks yang diambil dari baris ke-$\mathbf{r_1}$ sampai baris ke-$\mathbf{r_2}$ dan kolom ke-$\mathbf{c_1}$ sampai kolom ke-$\mathbf{c_2}$. Berikut adalah submatriks dari matriks diatas yang diambil dari baris ke-3 sampai baris ke-4, dan kolom ke-2 sampai kolom ke-3.

$$\begin{pmatrix} 3 & 99 \\ 11 & 7 \end{pmatrix}$$

Submatriks tersebut didapat dari bagian matriks yang ditandai dengan kotak berwarna merah.

$$\begin{pmatrix} 5 & 3 & 2 \\ 8 & 1 & 3 \\ 7 & 3 & 99 \\ 100 & 11 & 7 \end{pmatrix}$$

Untuk latihan kali ini, Anda akan membuat program yang membaca matriks pada suatu *file* yang diberikan, kemudian menampilkan submatriks dari matriks tersebut ke layar. Nama *file*, batas baris dan batas kolom pada submatriks akan diberikan sebagai masukan program.

Matriks akan disimpan pada suatu *file*. Terdapat beberapa *file* yang berisi matriks yang mempunyai ukuran yang berbeda-beda. Setiap baris akan dipisahkan oleh baris pada *file* (atau karakter *newline* / '\n') dan setiap kolom akan dipisahkan oleh spasi.

Anda dapat mengunduh *file* matriks dengan mengklik tautan ini (jangan lupa untuk memindahkan seluruh *file* ke dalam folder yang sama dengan program Anda!). Anda dapat membuka *file* tersebut dengan aplikasi *text editor* apapun (Notepad pada Windows tidak disarankan, Anda bisa menggunakan WordPad/Spyder/Notepad++). Berikut informasi ukuran matriks pada setiap *file*:

- `input1`: Matriks berukuran 4 x 3, matriks sama dengan matriks contoh di atas.
- `input2`: Matriks berukuran 6 x 8.
- `input3`: Matriks berukuran 10 x 22.
- `input4`: Matriks berukuran 50 x 50, setiap bilangan bernilai [0, 9] agar mudah dibaca.

Program Anda akan mendapatkan poin berdasarkan kriteria berikut:

- Program Anda berjalan sesuai dengan yang diminta oleh soal. (65 poin)
- **Jangan lupa untuk melakukan melakukan *close* pada *file*!** (10 poin)
- Program Anda menerima batas baris dan batas kolom dengan ***1-based indexing***, yang artinya baris pertama adalah baris ke-1 dan kolom pertama adalah kolom ke-1. (10 poin)
- Program Anda akan berhenti dan **menampilkan "Rentang submatriks tidak valid!" tanpa keluaran apapun selain itu** apabila batas baris/kolom yang diberikan masukan berada diluar ukuran matriks. (5 poin)
    - Anda harus menghitung baris dan kolom dari matriks ada file masukan.
    - Poin tidak dihitung apabila Anda melakukan prosedur *if* berdasarkan nama file untuk mendapatkan ukuran matriks.

Berikut adalah contoh interaksi program. Anda dibebaskan untuk menentukan format masukan sendiri. Tulisan tebal berwarna biru adalah masukan Anda. `r1`, `r2`, `c1`, `c2` masing-masing merupakan nilai $\mathbf{r_1}$, $\mathbf{r_2}$ $\mathbf{c_1}$ $\mathbf{c_2}$ yang telah dijelaskan diatas.

```
Masukkan nama file: input1
Masukkan r1: 3
Masukkan r2: 4
Masukkan c1: 2
Masukkan c2: 3
Submatriks:
3 99
11 7
```

Contoh di atas merupakan contoh submatriks yang telah dijelaskan diatas.

Contoh lain:

```
Masukkan nama file: input2
Masukkan r1: 1
Masukkan r2: 6
Masukkan c1: 1
Masukkan c2: 8
Submatriks:
39 3 3 14 27 22 8 29
23 6 28 45 4 22 24 28
35 42 39 12 24 22 1 22
14 35 35 2 48 35 26 43
17 30 35 11 31 46 21 5
22 19 35 33 41 32 44 6
```

Contoh lain:

```
Masukkan nama file: input3
Masukkan r1: 5
Masukkan r2: 7
Masukkan c1: 4
Masukkan c2: 16
Submatriks:
24 10 13 23 34 10 8 22 49 31 5 43 12
10 11 10 6 2 34 34 25 13 40 29 31 3
25 13 22 22 25 10 23 21 33 31 35 31 39
```

Contoh lain:

```
Masukkan nama file: input4
Masukkan r1: 1
Masukkan r2: 6
Masukkan c1: 1
Masukkan c2: 51
Rentang submatriks tidak valid!
```

Perhatikan bahwa batas kolom berada diluar ukuran matriks karena ukuran matriks pada *file* `input4` adalah 50 x 50.

Latihan #15: Bantuin Dong, Kak! (latihan15.py, skor: +10)

Konsep penting: variables, mutabie types, function, debugging!

Sekarang, bayangkan saat pagi tadi, teman baru/adik Anda yang bernama Namron tiba-tiba menanyakan Anda (yang baru saja bangun) tentang tugas pemrogramannya:

Ternyata, Anda lupa untuk menjawab pertanyaannya sampai sore ini! Anda sekarang ingin mengetahui apa *bug* dari program yang diberikan:

```
def isi_list(pengen_isi, baris, kolom):
    namron_punya_cnt = 0
    for i in range(baris):
        if i % 2 == 0:
            for j in range(kolom):
                pengen_isi[i][j] = namron_punya_cnt
                namron_punya_cnt += 1
        else:
            for j in range(kolom - 1, -1, -1):
                pengen_isi[i][j] = namron_punya_cnt
                namron_punya_cnt += 1

def cetak_list(pengen_cetak, baris, kolom):
    for i in range(5):
        for j in range(7):
            print(namron_punya_list[i][j], end=' ')
        print()

namron_punya_list = [[0] * 7] * 5
isi_list(namron_punya_list, 5, 7)
cetak_list(namron_punya_list, 5, 7)
```

Keluaran yang diharapkan adalah sesuai apa yang dikatakan Namron, yaitu:

```
0 1 2 3 4 5 6
13 12 11 10 9 8 7
14 15 16 17 18 19 20
27 26 25 24 23 22 21
28 29 30 31 32 33 34
```

Untuk menjawab soal ini, Anda cukup meng-*comment* baris kode yang bermasalah, dan berikan *comment* mengapa baris tersebut bermasalah dengan jelas, kemudian ganti baris kode bermasalah tersebut dengan baris yang benar. Berikut adalah contoh jawaban yang valid, namun bukan jawaban yang benar:

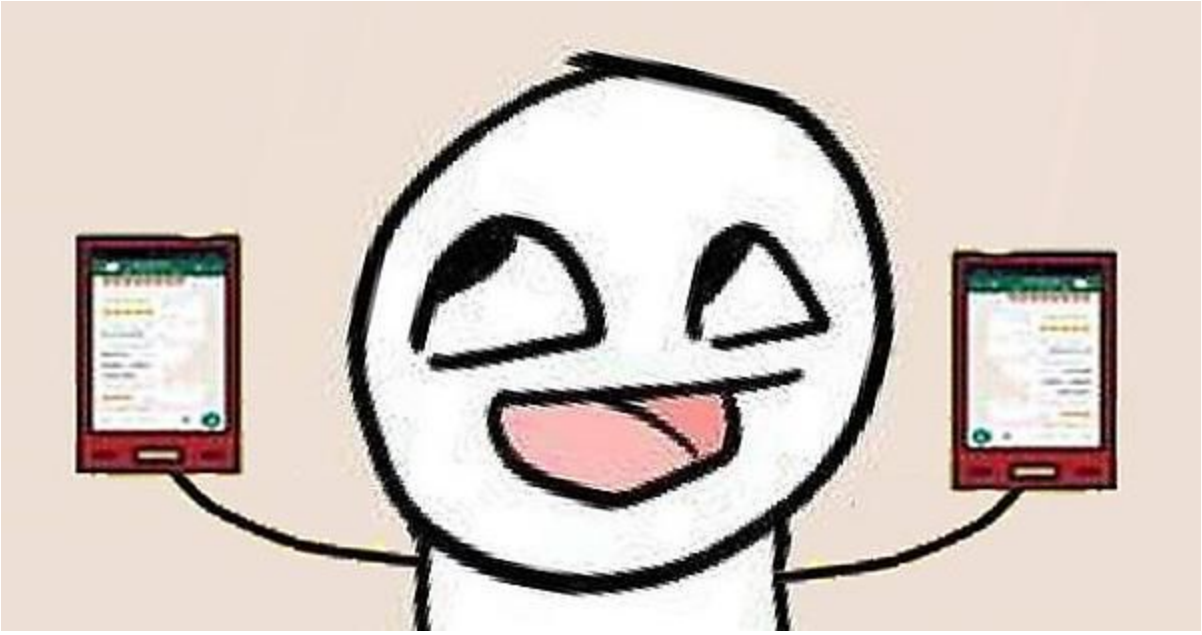
```
def isi_list(pengen_isi, baris, kolom):
    namron_punya_cnt = 0
    for i in range(baris):
        if i % 2 == 0:
            for j in range(kolom):
                pengen_isi[i][j] = namron_punya_cnt
                namron_punya_cnt += 1
        else:
            for j in range(kolom - 1, -1, -1):
                pengen_isi[i][j] = namron_punya_cnt
                # Bagian ini bermasalah (saya tidak suka angka 1)
                # namron_punya_cnt += 1
                # Berikut adalah kode yang benar
                namron_punya_cnt += 3

def cetak_list(pengen_cetak, baris, kolom):
    for i in range(5):
        for j in range(7):
            print(namron_punya_list[i][j], end=' ')
        print()

namron_punya_list = [[0] * 7] * 5
isi_list(namron_punya_list, 5, 7)
cetak_list(namron_punya_list, 5, 7)
```

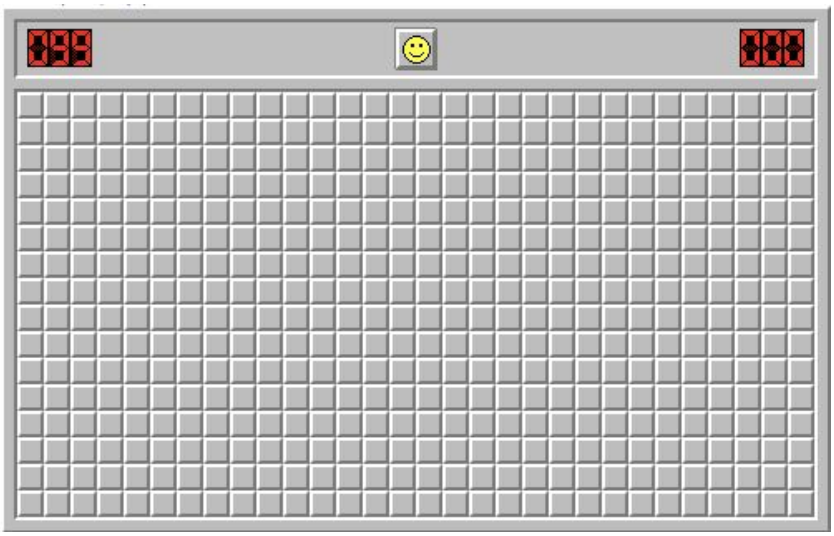
Peringatan: Apabila Anda mengumpulkan kode buatan Anda sendiri (bukan modifikasi dari kode di atas), Anda hanya akan mendapatkan 3 poin dari keseluruhan 10 poin. Anda juga tidak akan mendapatkan nilai apapun apabila Anda tidak memberikan penjelasan tentang baris kode yang bermasalah.

Behind The Scenes

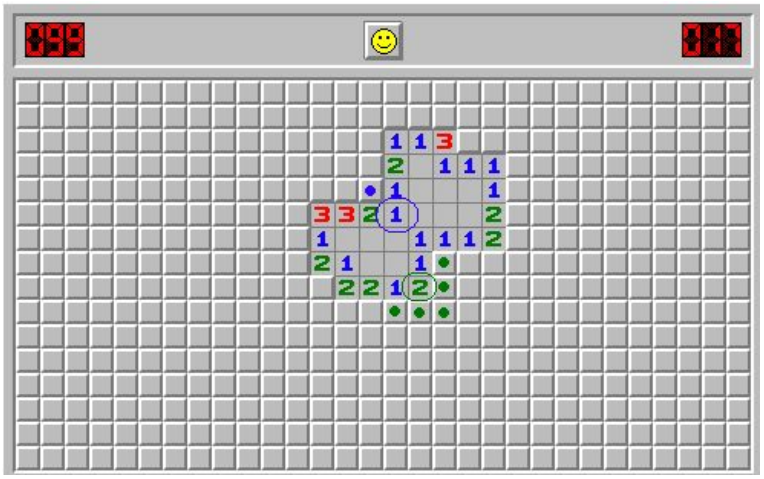


Challenge #4: *Minesweeper* (challenge4.py, skor: +0)

Apakah Anda tahu permainan *minesweeper*?



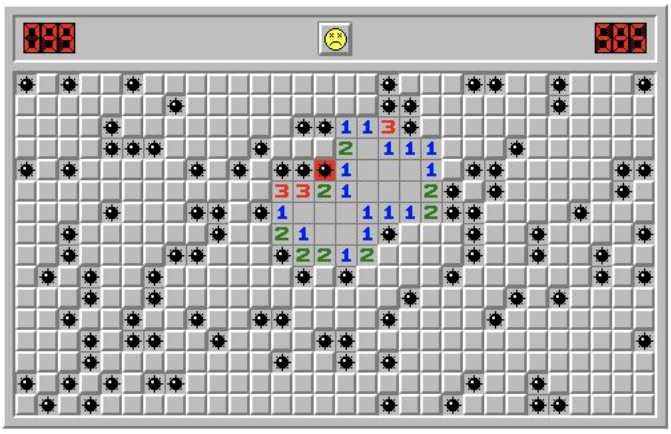
Permainan ini merupakan permainan klasik yang selalu ada sebagai *default game* di beberapa versi Windows XP, Vista, dan 7. Tujuan dari permainan tersebut sangat mudah, yaitu membuka seluruh petak yang ada kecuali petak yang berisi bom. Apabila kita membuka petak yang berisi bom, maka bom akan meledak dan permainan berakhir. Apabila kita membuka petak yang tidak berisi bom, maka petak tersebut akan terbuka dan menunjukkan suatu angka yang menandakan banyaknya bom yang tertanam disekitar petak tersebut (kiri, kanan, atas, bawah, kiri atas, kanan atas, kiri bawah, dan kanan bawah). Sebagai contoh, berikut adalah contoh petak yang terbuka;



Pada petak yang dilingkari biru, terlihat bahwa angka 1 menunjukkan bahwa terdapat 1 bom pada petak di sekitar petak tersebut (diberi titik biru). Dikarenakan hanya terdapat satu petak yang tidak terbuka, maka dapat dipastikan petak tersebut berisi bom.

Untuk petak yang dilingkari oleh warna hijau, terlihat bahwa terdapat dua bom di antara petak-petak di sekitar petak tersebut (diberi titik hijau). Kita tidak tahu petak mana yang berisi bom dengan hanya menggunakan informasi angka 2, kita perlu mencari informasi tambahan dari petak-petak di sekitar yang terbuka.

Tentu saja, apabila Anda membuka petak yang berisi bom, maka permainan berakhir dan Anda kalah.



Anda dapat mencoba permainan minesweeper pada tautan berikut agar Anda lebih paham (jangan lupa kerjakan soalnya!): http://minesweeperonline.com

Tugas Anda kali ini adalah membuat konfigurasi petak dari Minesweeper. Anda akan diberikan posisi dari setiap bom, kemudian Anda harus menentukan angka yang cocok untuk petak-petak yang bukan berisi bom. Anda harus menampilkan konfigurasi petak beserta angkanya ke layar komputer Anda. Sebagai contoh, Apabila Anda diberikan petak berukuran 5x5 berikut:

```
.....
.***.
.....
.....
*....
```

dengan tanda '.' (tanpa petik) menandakan petak yang tidak berisi bom dan '*' (tanpa petik) menandakan petak yang berisi bom, maka konfigurasi petak yang tepat adalah sebagai berikut:

```
12321
```

```
1***1
12321
11...
*1...
```

Perhatikan bahwa posisi bom tetap ditandai dengan '*', dan petak yang tidak mempunyai bom di sekitarnya ditandai dengan '.' (petak kosong).

Ketentuan program yang akan Anda buat adalah sebagai berikut:

1. Program Anda akan menerima informasi tentang banyaknya baris dan kolom dari petak permainan yang akan diberikan, dan informasi tentang petak tersebut termasuk posisi mana saja yang berisi bom.
2. Program Anda harus menampilkan petak dan angkanya dengan format berikut:
    a. Banyak baris dan banyak kolom harus sama dengan petak awal (masukan).
    b. Apabila petak tersebut berisi bom, keluarkan simbol '*' (tanpa tanda petik).
    c. Apabila petak tersebut tidak berisi bom, dan tidak ada bom di sekitar petak tersebut, keluarkan simbol '.' (tanpa tanda petik).
    d. Apabila petak tersebut tidak berisi bom, dan ada bom di sekitar petak tersebut, keluarkan sebuah angka yaitu banyaknya bom di sekitar petak tersebut.

Anda dibebaskan untuk menentukan format masukan-keluaran program Anda **asalkan mengikuti ketentuan di atas**.

Contoh masukan:

```
9 10
......**..
..**...*..
*.........
.***......
.........*
*........*
......*...
.*........
..*.*....*
```

Penjelasan: Petak di atas berukuran 8x10.

Contoh keluaran:

```
.12211**2.
12**113*2.
*4542.111.
2***1...11
23321...2*
*1...1112*
221..1*111
1*22121111
12*2*1..1*
```