Assignment 2

COMP9021, Trimester 2, 2019

1. General matter

1.1. **Aims.** The purpose of the assignment is to:

基于应用程序的期望行为设计和实现接口

- design and implement an interface based on the desired behaviour of an application program;
- practice the use of Python syntax;

语法

- develop problem solving skills.

1.2. **Submission.** Your program will be stored in a file named `polygons.py`. After you have developed and tested your program, upload it using Ed (unless you worked directly in Ed). Assignments can be submitted more than once; the last version is marked. Your assignment is due by August 11, 11:59pm.

1.3. **Assessment.** The assignment is worth 10 marks. It is going to be tested against a number of input files. For each test, the automarking script will let your program run for 30 seconds.

Late assignments will be penalised: the mark for a late submission will be the minimum of the awarded mark and 10 minus the number of full and partial days that have elapsed from the due date.

1.4. **Reminder on plagiarism policy.** You are permitted, indeed encouraged, to discuss ways to solve the assignment with other people. Such discussions must be in terms of algorithms, not code. But you must implement the solution on your own. Submissions are routinely scanned for similarities that occur when students copy and modify other people's work, or work very closely together on a single implementation. Severe penalties apply.

2. General presentation

设计并实施一个程序

You will design and implement a program that will

提取并分析（简单）多边形的各种特征，将它们的轮廓编码并存储在文件中

- extract and analyse the various characteristics of (simple) *polygons*, their contours being coded and stored in a file, and

要么显示那些特征：周长，面积，凸度，保持多边形不变的旋转次数，以及深度（最长的包围多边形链的长度）

- – either display those characteristics: perimeter, area, convexity, number of rotations that keep the polygon invariant, and depth (the length of the longest chain of enclosing polygons)
  – or output some Latex code, to be stored in a file, from which a pictorial representation of the polygons can be produced, coloured in a way which is proportional to their area.

或者输出一些Latex代码，存储在一个文件中，从中可以生成多边形的图形表示，并以与其面积成比例的方式着色

Call *encoding* any 2-dimensional grid of size between between 2×2 and 50×50 (both dimensions can be different) all of whose elements are either 0 or 1.

Callencodingany大小介于2×2和50×50之间的二维网格（两个维度可以不同）所有元素都是0或1

Call *neighbour* of a member m of an encoding any of the at most eight members of the grid whose value is 1 and each of both indexes differs from m's corresponding index by at most 1. Given a particular encoding, we inductively define for all natural numbers d the *set of polygons of depth* d (for this encoding) as follows. Let a natural number d be given, and suppose that for all $d' < d$, the set of polygons of depth d' has been defined. Change in the encoding all 1's that determine those polygons to 0. Then the set of polygons of depth d is defined as the set of polygons which can be obtained from that encoding by connecting 1's with some of their neighbours in such a way that we obtain a **maximal** polygon (that is, a polygon which is not included in any other polygon obtained from that encoding by connecting 1's with some of their neighbours).

归纳定义　　给定一个特定的编码，我们引导定义所有自然数的深度为多边形的集合（对于这种编码）如下

调用编码大小在2×2和50×50之间的二维网格（两个维度可以是不同的）所有其元素都是0或1.编码成员的一个成员的最近八个成员中的任何一个，其值是 1和两个索引中的每一个都与m的对应索引相差至多1.给定一个特定的编码，我们将所有自然数量的深度为（对于这种编码）的多边形的集合定义如下。 给出解剖数，并假设对于所有d'<d，已经定义了深度为多边形的集合。在编码中改变所有1，将这些多边形确定为0.然后将深度多边形的集合定义为集合 多边形可以通过连接1和它们的一些高度来获得，这样我们就可以获得极大多边形（也就是说，通过将1与它们的一些邻居连接起来，从那个编码中获得的任何其他多边形中都不包含多边形）

3. Examples

3.1. **First example.** The file `polys_1.txt` has the following contents:

```
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
```

Here is a possible interaction:

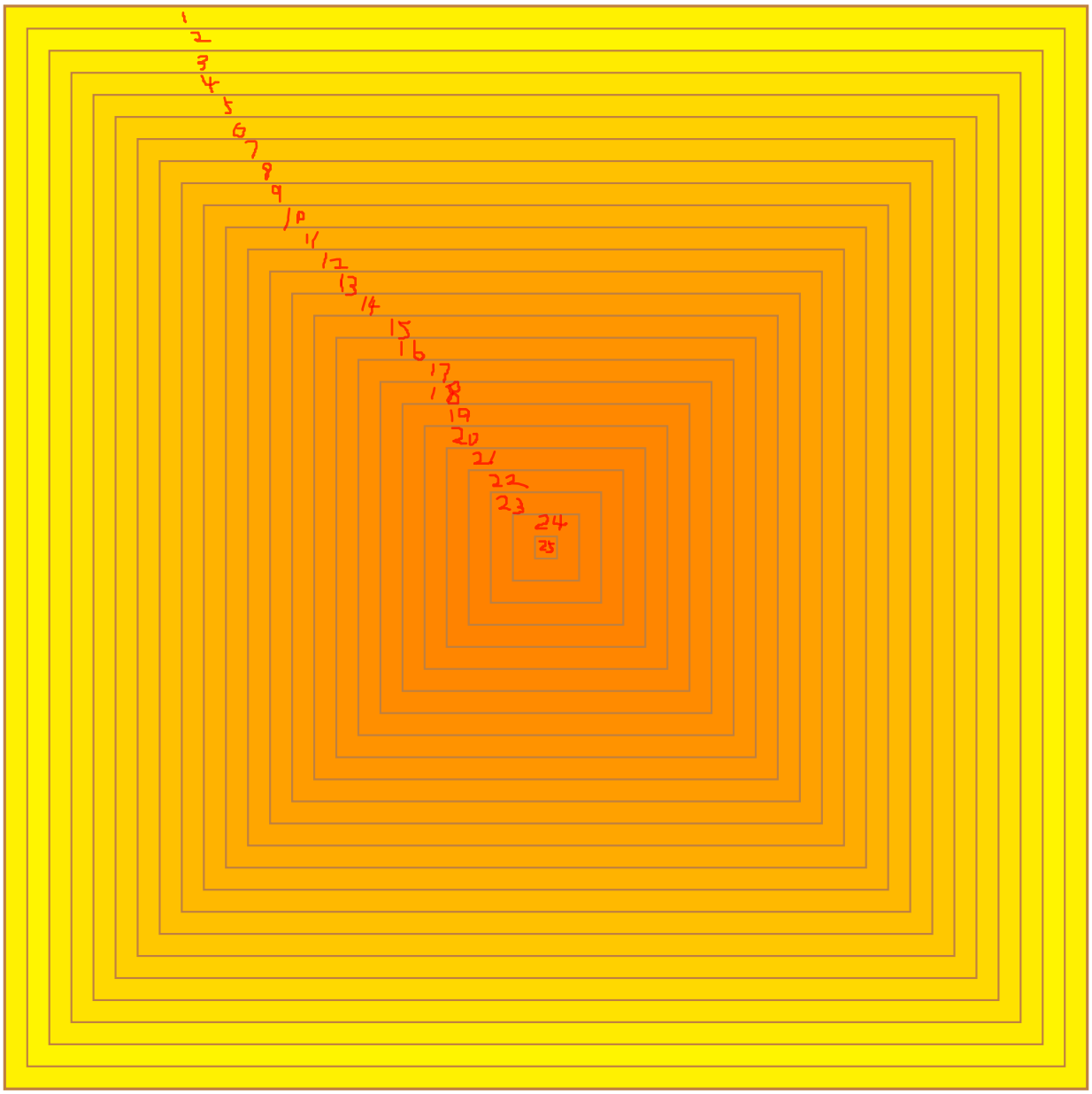
```
$ python3
...
>>> from polygons import *
>>> polys = Polygons('polys_1.txt')
>>> polys.analyse()
Polygon 1:
    Perimeter: 78.4
    Area: 384.16
    Convex: yes
    Nb of invariant rotations: 4
    Depth: 0
Polygon 2:
    Perimeter: 75.2
    Area: 353.44
    Convex: yes
    Nb of invariant rotations: 4
    Depth: 1
Polygon 3:
    Perimeter: 72.0
    Area: 324.00
    Convex: yes
    Nb of invariant rotations: 4
    Depth: 2
Polygon 4:
    Perimeter: 68.8
    Area: 295.84
    Convex: yes
    Nb of invariant rotations: 4
    Depth: 3
Polygon 5:
    Perimeter: 65.6
    Area: 268.96
    Convex: yes
    Nb of invariant rotations: 4
    Depth: 4
Polygon 6:
    Perimeter: 62.4
    Area: 243.36
    Convex: yes
    Nb of invariant rotations: 4
    Depth: 5
Polygon 7:
    Perimeter: 59.2
    Area: 219.04
    Convex: yes
    Nb of invariant rotations: 4
    Depth: 6
Polygon 8:
    Perimeter: 56.0
    Area: 196.00
    Convex: yes
    Nb of invariant rotations: 4
```

周长
面积
凸面体
不变的旋转

```
    Depth: 7
Polygon 9:
    Perimeter: 52.8
    Area: 174.24
    Convex: yes
    Nb of invariant rotations: 4
    Depth: 8
Polygon 10:
    Perimeter: 49.6
    Area: 153.76
    Convex: yes
    Nb of invariant rotations: 4
    Depth: 9
Polygon 11:
    Perimeter: 46.4
    Area: 134.56
    Convex: yes
    Nb of invariant rotations: 4
    Depth: 10
Polygon 12:
    Perimeter: 43.2
    Area: 116.64
    Convex: yes
    Nb of invariant rotations: 4
    Depth: 11
Polygon 13:
    Perimeter: 40.0
    Area: 100.00
    Convex: yes
    Nb of invariant rotations: 4
    Depth: 12
Polygon 14:
    Perimeter: 36.8
    Area: 84.64
    Convex: yes
    Nb of invariant rotations: 4
    Depth: 13
Polygon 15:
    Perimeter: 33.6
    Area: 70.56
    Convex: yes
    Nb of invariant rotations: 4
    Depth: 14
Polygon 16:
    Perimeter: 30.4
    Area: 57.76
    Convex: yes
    Nb of invariant rotations: 4
    Depth: 15
Polygon 17:
    Perimeter: 27.2
    Area: 46.24
    Convex: yes
    Nb of invariant rotations: 4
```

```
    Depth: 16
Polygon 18:
    Perimeter: 24.0
    Area: 36.00
    Convex: yes
    Nb of invariant rotations: 4
    Depth: 17
Polygon 19:
    Perimeter: 20.8
    Area: 27.04
    Convex: yes
    Nb of invariant rotations: 4
    Depth: 18
Polygon 20:
    Perimeter: 17.6
    Area: 19.36
    Convex: yes
    Nb of invariant rotations: 4
    Depth: 19
Polygon 21:
    Perimeter: 14.4
    Area: 12.96
    Convex: yes
    Nb of invariant rotations: 4
    Depth: 20
Polygon 22:
    Perimeter: 11.2
    Area: 7.84
    Convex: yes
    Nb of invariant rotations: 4
    Depth: 21
Polygon 23:
    Perimeter: 8.0
    Area: 4.00
    Convex: yes
    Nb of invariant rotations: 4
    Depth: 22
Polygon 24:
    Perimeter: 4.8
    Area: 1.44
    Convex: yes
    Nb of invariant rotations: 4
    Depth: 23
Polygon 25:
    Perimeter: 1.6
    Area: 0.16
    Convex: yes
    Nb of invariant rotations: 4
    Depth: 24
>>> polys.display()
```

The effect of executing `polys.display()` is to produce a file named `polys_1.tex` that can be given as argument to `pdflatex` to produce a file named `polys_1.pdf` that views as follows.



3.2. **Second example.** The file polys_2.txt has the following contents:

```
00000000000000000000000000000000000000000000000000
01111111111111111111111111111111111111111111111110
00111111111111111111111111111111111111111111111100
00011111111111111111111111111111111111111111111000
01001111111111111111111111111111111111111111110010
01100111111111111111111111111111111111111111100110
01110011111111111111111111111111111111111111001110
01111001111111111111111111111111111111111110011110
01111100111111111111111111111111111111111100111110
01111110011111111111111111111111111111111001111110
01111111001111111111111111111111111111110011111110
01111111100111111111111111111111111111100111111110
01111111110011111111111111111111111111001111111110
01111111111001111111111111111111111110011111111110
01111111111100111111111111111111111100111111111110
01111111111110011111111111111111111001111111111110
01111111111111001111111111111111110011111111111110
01111111111111100111111111111111100111111111111110
01111111111111110011111111111111001111111111111110
01111111111111111001111111111110011111111111111110
01111111111111111100111111111100111111111111111110
01111111111011111111110011001111111111011111111110
01111111111111111111100111100111111111111111111110
01111111111111111111001111001111111111111111111110
01111111111111111110011111100111111111111111111110
01111111111111111100111111110011111111111111111110
01111111111111111001111111111001111111111111111110
01111111111111110011111111111100111111111111111110
01111111111111100111111111111110011111111111111110
01111111111111001111111111111111001111111111111110
01111111111110011111111111111111100111111111111110
01111111111100111111111111111111110011111111111110
01111111111001111111111111111111111001111111111110
01111111110011111111111111111111111100111111111110
01111111100111111111111111111111111110011111111110
01111111001111111111111111111111111111001111111110
01111110011111111111111111111111111111100111111110
01111100111111111111111111111111111111110011111110
01111001111111111111111111111111111111111001111110
01110011111111111111111111111111111111111100111110
01100111111111111111111111111111111111111110011110
01001111111111111111111111111111111111111111001110
00011111111111111111111111111111111111111111000110
00111111111111111111111111111111111111111111100010
01111111111111111111111111111111111111111111110000
00000000000000000000000000000000000000000000000000
```

Here is a possible interaction:

```
$ python3
...
>>> from polygons import *
>>> polys = Polygons('polys_2.txt')
>>> polys.analyse()
Polygon 1:
    Perimeter: 37.6 + 92*sqrt(.32)
    Area: 176.64
    Convex: no
    Nb of invariant rotations: 2
    Depth: 0
Polygon 2:
    Perimeter: 17.6 + 42*sqrt(.32)
    Area: 73.92
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 1
Polygon 3:
    Perimeter: 16.0 + 38*sqrt(.32)
    Area: 60.80
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 2
Polygon 4:
    Perimeter: 16.0 + 40*sqrt(.32)
    Area: 64.00
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 0
Polygon 5:
    Perimeter: 14.4 + 34*sqrt(.32)
    Area: 48.96
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 3
Polygon 6:
    Perimeter: 16.0 + 40*sqrt(.32)
    Area: 64.00
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 0
Polygon 7:
    Perimeter: 12.8 + 30*sqrt(.32)
    Area: 38.40
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 4
Polygon 8:
    Perimeter: 14.4 + 36*sqrt(.32)
    Area: 51.84
    Convex: yes
    Nb of invariant rotations: 1
```

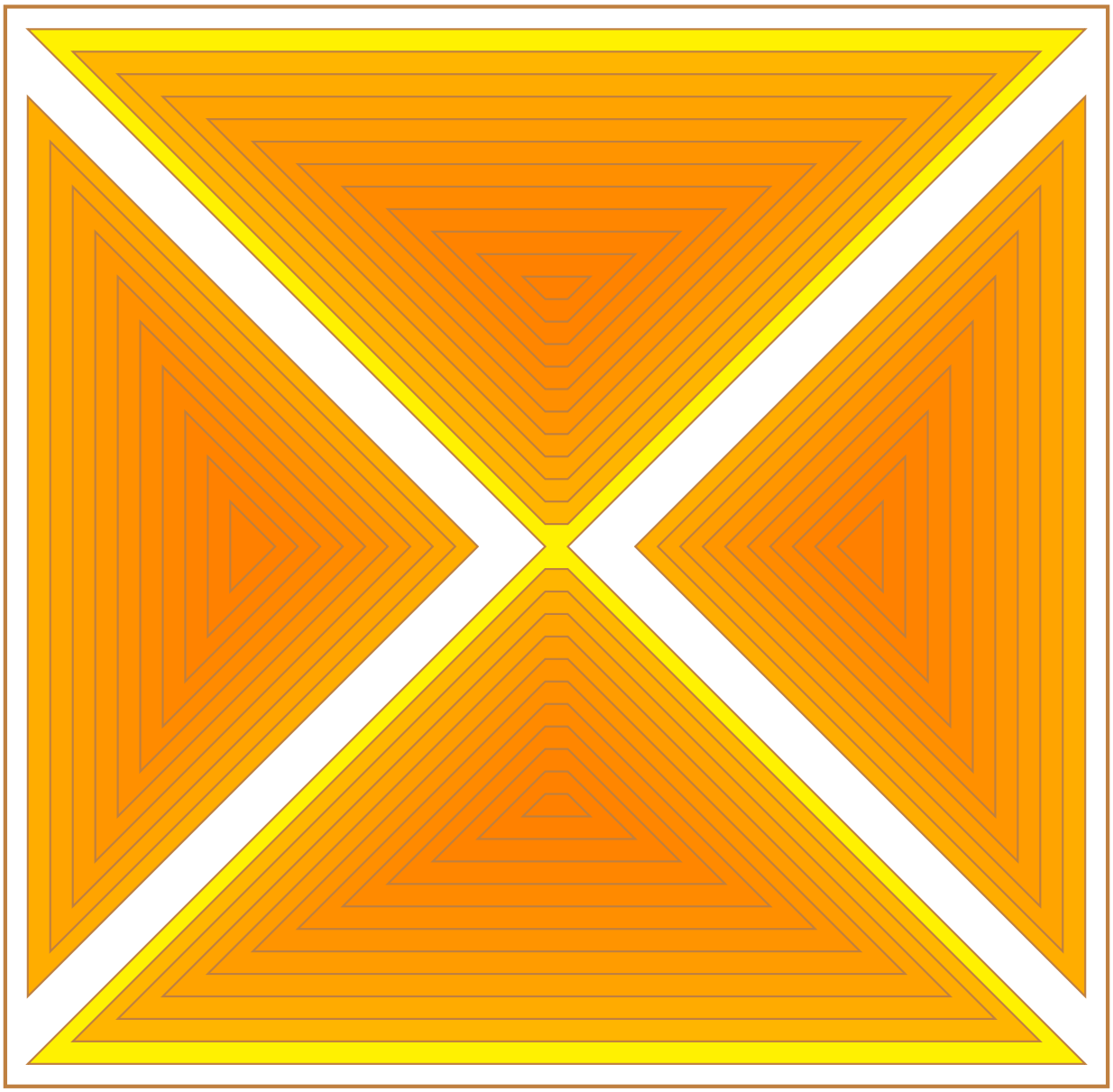
```
    Depth: 1
Polygon 9:
    Perimeter: 11.2 + 26*sqrt(.32)
    Area: 29.12
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 5
Polygon 10:
    Perimeter: 14.4 + 36*sqrt(.32)
    Area: 51.84
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 1
Polygon 11:
    Perimeter: 9.6 + 22*sqrt(.32)
    Area: 21.12
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 6
Polygon 12:
    Perimeter: 12.8 + 32*sqrt(.32)
    Area: 40.96
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 2
Polygon 13:
    Perimeter: 8.0 + 18*sqrt(.32)
    Area: 14.40
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 7
Polygon 14:
    Perimeter: 12.8 + 32*sqrt(.32)
    Area: 40.96
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 2
Polygon 15:
    Perimeter: 6.4 + 14*sqrt(.32)
    Area: 8.96
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 8
Polygon 16:
    Perimeter: 11.2 + 28*sqrt(.32)
    Area: 31.36
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 3
Polygon 17:
    Perimeter: 4.8 + 10*sqrt(.32)
    Area: 4.80
    Convex: yes
    Nb of invariant rotations: 1
```

```
    Depth: 9
Polygon 18:
    Perimeter: 11.2 + 28*sqrt(.32)
    Area: 31.36
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 3
Polygon 19:
    Perimeter: 3.2 + 6*sqrt(.32)
    Area: 1.92
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 10
Polygon 20:
    Perimeter: 9.6 + 24*sqrt(.32)
    Area: 23.04
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 4
Polygon 21:
    Perimeter: 1.6 + 2*sqrt(.32)
    Area: 0.32
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 11
Polygon 22:
    Perimeter: 9.6 + 24*sqrt(.32)
    Area: 23.04
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 4
Polygon 23:
    Perimeter: 8.0 + 20*sqrt(.32)
    Area: 16.00
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 5
Polygon 24:
    Perimeter: 8.0 + 20*sqrt(.32)
    Area: 16.00
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 5
Polygon 25:
    Perimeter: 6.4 + 16*sqrt(.32)
    Area: 10.24
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 6
Polygon 26:
    Perimeter: 6.4 + 16*sqrt(.32)
    Area: 10.24
    Convex: yes
    Nb of invariant rotations: 1
```

```
    Depth: 6
Polygon 27:
    Perimeter: 4.8 + 12*sqrt(.32)
    Area: 5.76
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 7
Polygon 28:
    Perimeter: 4.8 + 12*sqrt(.32)
    Area: 5.76
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 7
Polygon 29:
    Perimeter: 3.2 + 8*sqrt(.32)
    Area: 2.56
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 8
Polygon 30:
    Perimeter: 3.2 + 8*sqrt(.32)
    Area: 2.56
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 8
Polygon 31:
    Perimeter: 1.6 + 4*sqrt(.32)
    Area: 0.64
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 9
Polygon 32:
    Perimeter: 1.6 + 4*sqrt(.32)
    Area: 0.64
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 9
Polygon 33:
    Perimeter: 17.6 + 42*sqrt(.32)
    Area: 73.92
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 1
Polygon 34:
    Perimeter: 16.0 + 38*sqrt(.32)
    Area: 60.80
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 2
Polygon 35:
    Perimeter: 14.4 + 34*sqrt(.32)
    Area: 48.96
    Convex: yes
    Nb of invariant rotations: 1
```

```
    Depth: 3
Polygon 36:
    Perimeter: 12.8 + 30*sqrt(.32)
    Area: 38.40
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 4
Polygon 37:
    Perimeter: 11.2 + 26*sqrt(.32)
    Area: 29.12
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 5
Polygon 38:
    Perimeter: 9.6 + 22*sqrt(.32)
    Area: 21.12
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 6
Polygon 39:
    Perimeter: 8.0 + 18*sqrt(.32)
    Area: 14.40
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 7
Polygon 40:
    Perimeter: 6.4 + 14*sqrt(.32)
    Area: 8.96
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 8
Polygon 41:
    Perimeter: 4.8 + 10*sqrt(.32)
    Area: 4.80
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 9
Polygon 42:
    Perimeter: 3.2 + 6*sqrt(.32)
    Area: 1.92
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 10
Polygon 43:
    Perimeter: 1.6 + 2*sqrt(.32)
    Area: 0.32
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 11
>>> polys.display()
```

The effect of executing `polys.display()` is to produce a file named `polys_2.tex` that can be given as argument to `pdflatex` to produce a file named `polys_2.pdf` that views as follows.



3.3. **Third example.** The file polys_3.txt has the following contents:

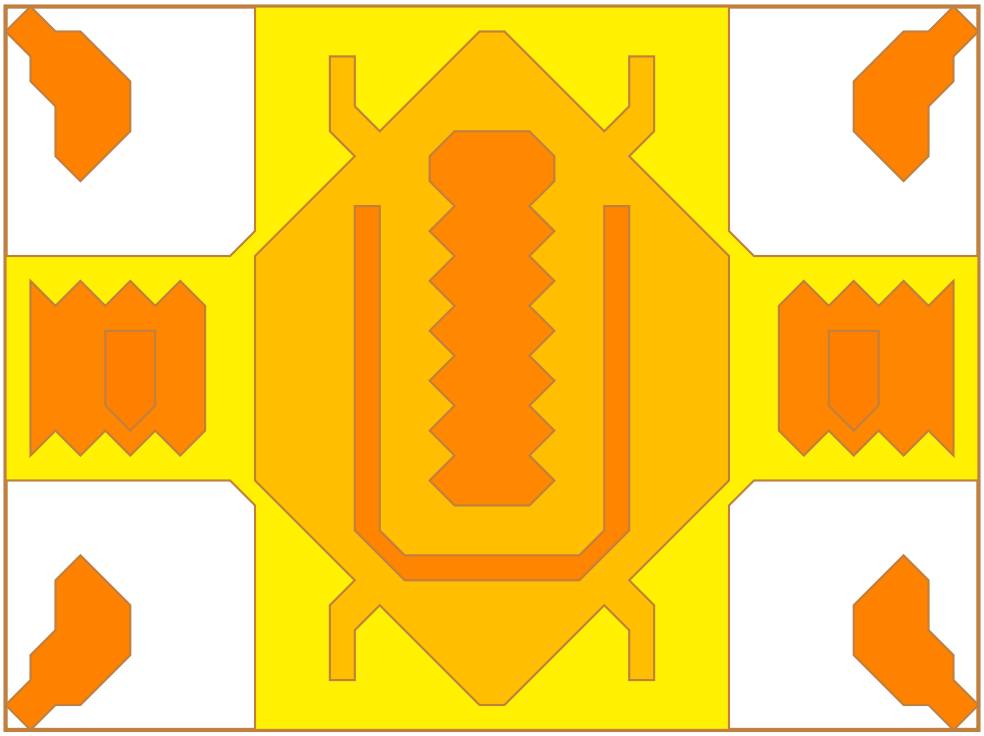
```
0 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 0
1 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 1
0 1 0 0 1 0 0 0 0 1 0 0 1 1 0 0 0 1 0 0 1 0 0 0 1 1 0 0 1 0 0 0 0 0 1 0 0 1 0
0 1 0 0 0 1 0 0 0 0 1 0 0 1 1 0 0 1 0 0 0 0 1 0 0 1 1 0 0 1 0 0 0 0 1 0 0 0 1 0
0 0 1 0 0 1 0 0 0 0 1 0 0 1 1 0 1 0 0 0 0 0 0 1 0 1 1 0 0 1 0 0 0 0 1 0 0 1 0 0
0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1 0 0 1 1 1 1 0 0 1 0 1 0 0 1 0 0 0 0 1 0 0 1 0 0
0 0 1 0 1 0 0 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0 1 0 1 0 0
0 0 0 1 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 1 0 1 0 1 1 0 0 1 0 0 1 0 0 1 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 1 0 0 0 0 1 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 0 0 1 0 0 1 0 0 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1
1 1 0 1 0 1 0 1 0 0 1 0 0 0 1 1 0 1 0 0 0 0 1 0 1 1 0 0 0 1 0 0 1 0 1 0 1 0 1 1
1 1 1 0 1 0 1 0 1 0 1 0 0 0 1 1 0 0 1 0 0 1 0 0 1 1 0 0 0 1 0 1 0 1 0 1 0 1 1 1
1 1 0 0 1 1 1 0 1 0 1 0 0 0 1 1 0 1 0 0 0 0 1 0 1 1 0 0 0 1 0 1 0 1 1 1 0 0 1 1
1 1 0 0 1 0 1 0 1 0 1 0 0 0 1 1 0 0 1 0 0 1 0 0 1 1 0 0 0 1 0 1 0 1 0 1 0 0 1 1
1 1 0 0 1 0 1 0 1 0 1 0 0 0 1 1 0 1 0 0 0 0 1 0 1 1 0 0 0 1 0 1 0 1 0 1 0 0 1 1
1 1 0 0 1 0 1 0 1 0 1 0 0 0 1 1 0 0 1 0 0 1 0 0 1 1 0 0 0 1 0 1 0 1 0 1 0 0 1 1
1 1 1 0 1 1 1 0 1 0 1 0 0 0 1 1 0 1 0 0 0 0 1 0 1 1 0 0 0 1 0 1 0 1 1 1 0 1 1 1
1 1 0 1 0 1 0 1 0 0 1 0 0 0 1 1 0 0 1 0 0 1 0 0 1 1 0 0 0 1 0 0 1 0 1 0 1 0 1 1
1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 0 1 0 0 0 0 1 0 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 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 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 1 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 1 0 0 1 0 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 0 0 0 0 0 0 1 0 0 0
0 0 1 0 1 0 0 0 0 0 1 0 0 0 1 0 1 1 1 1 1 1 1 1 1 0 1 0 0 0 1 0 0 0 0 0 1 0 1 0 0
0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 1 0 0 1 0 0
0 0 1 0 0 1 0 0 0 0 1 0 0 1 1 0 1 0 0 0 0 0 0 1 0 1 1 0 0 1 0 0 0 0 1 0 0 1 0 0
0 1 0 0 0 1 0 0 0 0 1 0 0 1 1 0 0 1 0 0 0 0 1 0 0 1 1 0 0 1 0 0 0 0 1 0 0 0 1 0
0 1 0 0 1 0 0 0 0 0 1 0 0 1 1 0 0 0 1 0 0 1 0 0 0 1 1 0 0 1 0 0 0 0 0 1 0 0 1 0
1 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 1
0 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 0
```

Here is a possible interaction:

```
$ python3
...
>>> from polygons import *
>>> polys = Polygons('polys_3.txt')
>>> polys.analyse()
Polygon 1:
    Perimeter: 2.4 + 9*sqrt(.32)
    Area: 2.80
    Convex: no
    Nb of invariant rotations: 1
    Depth: 0
Polygon 2:
    Perimeter: 51.2 + 4*sqrt(.32)
    Area: 117.28
    Convex: no
    Nb of invariant rotations: 2
    Depth: 0
Polygon 3:
    Perimeter: 2.4 + 9*sqrt(.32)
    Area: 2.80
    Convex: no
    Nb of invariant rotations: 1
    Depth: 0
Polygon 4:
    Perimeter: 17.6 + 40*sqrt(.32)
    Area: 59.04
    Convex: no
    Nb of invariant rotations: 2
    Depth: 1
Polygon 5:
    Perimeter: 3.2 + 28*sqrt(.32)
    Area: 9.76
    Convex: no
    Nb of invariant rotations: 1
    Depth: 2
Polygon 6:
    Perimeter: 27.2 + 6*sqrt(.32)
    Area: 5.76
    Convex: no
    Nb of invariant rotations: 1
    Depth: 2
Polygon 7:
    Perimeter: 4.8 + 14*sqrt(.32)
    Area: 6.72
    Convex: no
    Nb of invariant rotations: 1
    Depth: 1
Polygon 8:
    Perimeter: 4.8 + 14*sqrt(.32)
    Area: 6.72
    Convex: no
    Nb of invariant rotations: 1
```

```
    Depth: 1
Polygon 9:
    Perimeter: 3.2 + 2*sqrt(.32)
    Area: 1.12
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 2
Polygon 10:
    Perimeter: 3.2 + 2*sqrt(.32)
    Area: 1.12
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 2
Polygon 11:
    Perimeter: 2.4 + 9*sqrt(.32)
    Area: 2.80
    Convex: no
    Nb of invariant rotations: 1
    Depth: 0
Polygon 12:
    Perimeter: 2.4 + 9*sqrt(.32)
    Area: 2.80
    Convex: no
    Nb of invariant rotations: 1
    Depth: 0
>>> polys.display()
```

The effect of executing `polys.display()` is to produce a file named `polys_3.tex` that can be given as argument to `pdflatex` to produce a file named `polys_3.pdf` that views as follows.



3.4. **Fourth example.** The file polys_4.txt has the following contents:

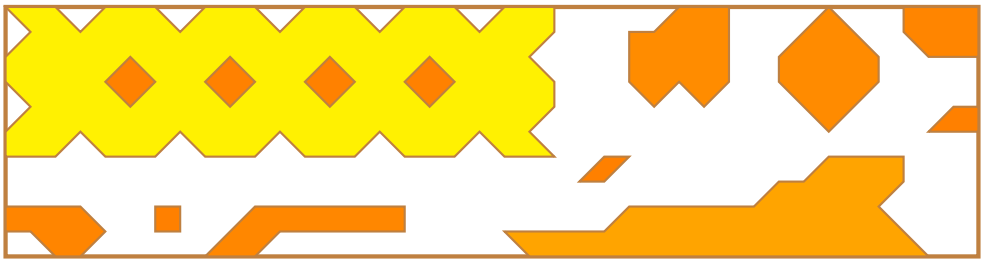
```
1    1  101     11 0  1 1            1 0 1      1 1011 10         1 1 1 0 000        1 1 1 0       00  1 001  11 1



01  0100010001000100010100                          110010010101001


100 0010 0              0 1            00 0 1 0 00          100          01000               100 0 1 01 0001011          1


1000101010101010101000100101010100010000
0100010001000100010000100010100010100011

100       1 0 0 0            10  0 0 1     00 0 1     00        01   010   000       0000     0 0 0 1       00 01      11


11101             1101110     1                  1           1                 011101110110000000111100
0000000000000000000000001100000011000100                                                  0

1                                  11100110011111110000000111111000      010000


110 01       0 1 1 0                                                  1011111100011111000000000001000

001  1000011       10           000000000                    11111111111111111                                    00
```

Here is a possible interaction:

```
$ python3
...
>>> from polygons import *
>>> polys = Polygons('polys_4.txt')
>>> polys.analyse()
Polygon 1:
    Perimeter: 11.2 + 28*sqrt(.32)
    Area: 18.88
    Convex: no
    Nb of invariant rotations: 2
    Depth: 0
Polygon 2:
    Perimeter: 3.2 + 5*sqrt(.32)
    Area: 2.00
    Convex: no
    Nb of invariant rotations: 1
    Depth: 0
Polygon 3:
    Perimeter: 0.8 + 8*sqrt(.32)
    Area: 1.92
    Convex: yes
    Nb of invariant rotations: 2
    Depth: 0
Polygon 4:
    Perimeter: 3.2 + 1*sqrt(.32)
    Area: 0.88
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 0
Polygon 5:
    Perimeter: 4*sqrt(.32)
    Area: 0.32
    Convex: yes
    Nb of invariant rotations: 4
    Depth: 1
Polygon 6:
    Perimeter: 4*sqrt(.32)
    Area: 0.32
    Convex: yes
    Nb of invariant rotations: 4
    Depth: 1
Polygon 7:
    Perimeter: 4*sqrt(.32)
    Area: 0.32
    Convex: yes
    Nb of invariant rotations: 4
    Depth: 1
Polygon 8:
    Perimeter: 4*sqrt(.32)
    Area: 0.32
    Convex: yes
    Nb of invariant rotations: 4
```

```
    Depth: 1
Polygon 9:
    Perimeter: 1.6 + 1*sqrt(.32)
    Area: 0.24
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 0
Polygon 10:
    Perimeter: 0.8 + 2*sqrt(.32)
    Area: 0.16
    Convex: yes
    Nb of invariant rotations: 2
    Depth: 0
Polygon 11:
    Perimeter: 12.0 + 7*sqrt(.32)
    Area: 5.68
    Convex: no
    Nb of invariant rotations: 1
    Depth: 0
Polygon 12:
    Perimeter: 2.4 + 3*sqrt(.32)
    Area: 0.88
    Convex: no
    Nb of invariant rotations: 1
    Depth: 0
Polygon 13:
    Perimeter: 1.6
    Area: 0.16
    Convex: yes
    Nb of invariant rotations: 4
    Depth: 0
Polygon 14:
    Perimeter: 5.6 + 3*sqrt(.32)
    Area: 1.36
    Convex: no
    Nb of invariant rotations: 1
    Depth: 0
>>> polys.display()
```

The effect of executing `polys.display()` is to produce a file named `polys_4.tex` that can be given as argument to `pdflatex` to produce a file named `polys_4.pdf` that views as follows.



4. Detailed description

预期；盼望

4.1. **Input.** The input is expected to consist of y_{dim} lines of x_{dim} 0's and 1's, where x_{dim} and y_{dim} are at least equal to 2 and at most equal to 50, with possibly lines consisting of spaces only that will be ignored and with possibly spaces anywhere on the lines with digits. If n is the x^{th} digit of the y^{th} line with digits, with $0 \leq x < x_{dim}$ and $0 \leq y < y_{dim}$, then n is to be associated with a point situated $x \times 0.4$ cm to the right and $y \times 0.4$ cm below an origin.

可能包含空格的行只会被忽略，并且可能在带有数字的行上的任何位置留空

4.2. **Output.** Consider executing from the Python prompt the statement `from polygons import *` followed by the statement `polys = Polygons(`*`some_filename`*`)`. In case *`some_filename`* does not exist in the working directory, then Python will raise a `FileNotFoundError` exception, that does not need to be caught. Assume that *`some_filename`* does exist (in the working directory). If the input is incorrect in that it does not contain only 0's and 1'a besides spaces, or in that it contains either too few or too many lines of digits, or in that some line of digits contains too many or too few digits, or in that two of its lines of digits do not contain the same number of digits, then the effect of executing `polys = Polygons(`*`some_filename`*`)` should be to generate a `PolygonsError` exception that reads

考虑从Python提示符执行多边形导入的语句*后跟语句polys = Polygons（some_filename）。 如果工作目录中不存在casesome_filename，那么Python将引发一个不需要捕获的FileNotFoundErrorexception。假设some_filename确实存在（在工作目录中）。 如果输入不正确，它除了空格之外不包含0和1'，或者它包含太少或太多的数字行，或者在某些数字行包含太多或太少的数字，或者在 它的两行数字不包含相同的数字位数，那么执行polys = Polygons（some_filename）的效果应该是生成一个读取的PolygonsError异常

```
Traceback (most recent call last):
...
polygons.PolygonsError: Incorrect input.
```

If the previous conditions hold but it is not possible to use all 1's in the input and make them the contours of polygons of depth d, for any natural number d, as defined in the general presentation, then the effect of executing `polys = Polygons(`*`some_filename`*`)` should be to generate a `PolygonsError` exception that reads

如果前面的条件成立但是不可能在输入中使用全1并且使它们成为深度的多边形的轮廓，对于任何自然数，如一般表示中所定义的那样，则执行polys =多边形（some_filename）的效果应该是 生成一个读取的多边形错误

```
Traceback (most recent call last):
...
polygons.PolygonsError: Cannot get polygons as expected.
```

无法得到预期的多边形

If the input is correct and it is possible to use all 1's in the input and make them the contours of polygons of depth d, for any natural number d, as defined in the general presentation, then executing the statement `polys = Polygons(`*`some_filename`*`)` followed by `polys.analyse()` should have the effect of outputting a first line that reads

如果输入是正确的，并且可以使用输入中的所有1并使它们成为深度为多边形的轮廓，对于任何自然数，如一般表示中所定义，则执行statementpolys = Polygons（some_filename），然后执行polys.analyse （）应该具有输出读取的第一行的效果

```
Polygon N:
```

with `N` an appropriate integer at least equal to 1 to refer to the `N`'th polygon listed in the order of polygons with highest point from smallest value of y to largest value of y, and for a given value of y, from smallest value of x to largest value of x, a second line that reads one of

```
Perimeter: a + b*sqrt(.32)
Perimeter: a
Perimeter: b*sqrt(.32)
```

with `a` an appropriate strictly positive floating point number with 1 digit after the decimal point and `b` an appropriate strictly positive integer, a third line that reads

```
Area: a
```

with `a` an appropriate floating point number with 2 digits after the decimal point, a fourth line that reads one of

```
Convex: yes
Convex: no
```

a fifth line that reads

```
Nb of invariant rotations: N
```

with `N` an appropriate integer at least equal to 1, and a sixth line that reads

```
Depth: N
```

with `N` an appropriate positive integer (possibly 0).

Pay attention to the expected format, including spaces.

If the input is correct and it is possible to use all 1's in the input and make them the contours of polygons of depth d, for any natural number d, as defined in the general presentation, then executing the statement `polys = Polygons(`*`some_filename`*`)` followed by `polys.display()` should have the effect of producing a file named *`some_filename`*`.tex` that can be given as argument to `pdflatex` to generate a file named *`some_filename`*`.pdf`. The provided examples will show you what *`some_filename`*`.tex` should contain.

- Polygons are drawn from lowest to highest depth, and for a given depth, the same ordering as previously described is used.
- The point that determines the polygon index is used as a starting point in drawing the line segments that make up the polygon, in a clockwise manner.
- A polygons's colour is determined by its area. The largest polygons are yellow. The smallest polygons are orange. Polygons in-between mix orange and yellow in proportion of their area. For instance, a polygon whose size is 25% the difference of the size between the largest and the smallest polygon will receive 25% of orange (and 75% of yellow). That proportion is computed as an integer. When the value is not an integer, it is rounded to the closest integer, with values of the form $z.5$ rounded up to $z + 1$.

Pay attention to the expected format, including spaces and blank lines. Lines that start with `%` are comments. The output of your program redirected to a file will be compared with the expected output saved in a file (of a different name of course) using the `diff` command. For your program to pass the associated test, `diff` should silently exit, which requires that the contents of both files be absolutely identical, character for character, including spaces and blank lines. Check your program on the provided examples using the associated .tex files, renaming them as they have the names of the files expected to be generated by your program.