Assignment 2

COMP9021, Session 1, 2017

1 General presentation

You will design and implement a program that will

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

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.

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).

1.1 Submission

Your programs will be stored in a file named `polygons.py`. After you have developed and tested your program, upload your files using Ed. Assignments can be submitted more than once: the last version is marked. Your assignment is due by May 7, 11:59pm.

1.2 Assessment

The assignment is worth 10 marks. the automarking script will allocate 30 seconds to each run of your program.

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.

The outputs of your programs should be **exactly** as indicated.

1.3 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 Examples

2.1 First example

Given a file named polys_1.txt whose contents is

```
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
```

your program when run as python3 polygons.py --file polys_1.txt should output

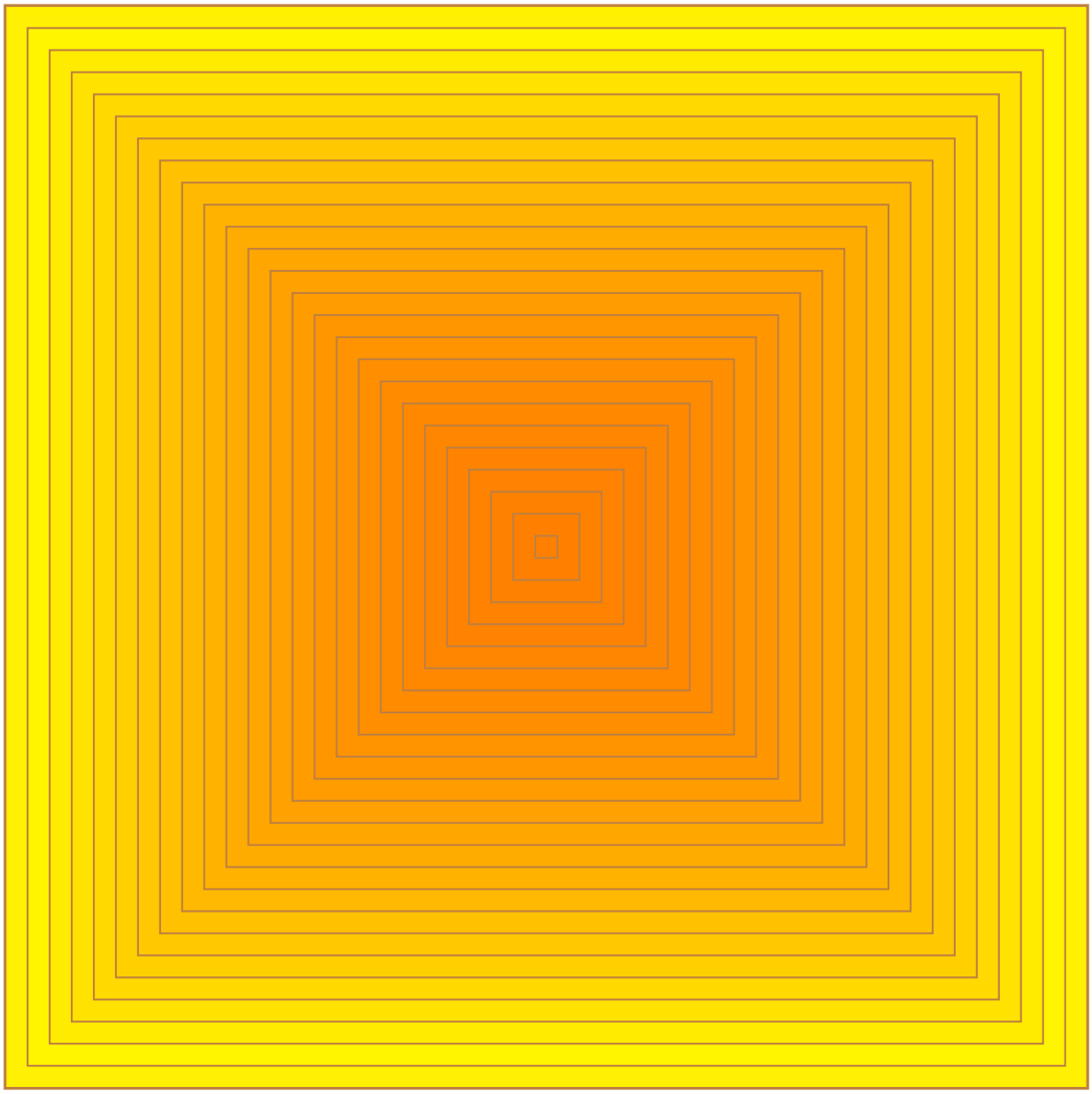
```
Polygon 1:
    Perimeter: 78.4
    Area: 384.16
    Convex: yes
    Nb of invariant rotations: 4
    Depth: 0
Polygon 2:
    Perimeter: 75.2
    Area: 353.44
    Convex: yes
    Nb of invariant rotations: 4
    Depth: 1
Polygon 3:
    Perimeter: 72.0
    Area: 324.00
    Convex: yes
    Nb of invariant 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
    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
```

and when run as `python3 polygons.py -print --file polys_1.txt` should produce some output saved in a file named `polys_1.tex`, which can be given as argument to `pdflatex` to produce a file named `polys_1.pdf` that views as follows.



2.2 Second example

Given a file named polys_2.txt whose contents is

```
0000000000000000000000000000000000000000000000000000
0111111111111111111111111111111111111111111111111110
0011111111111111111111111111111111111111111111111100
0001111111111111111111111111111111111111111111111000
0100111111111111111111111111111111111111111111110010
0110011111111111111111111111111111111111111111100110
0111001111111111111111111111111111111111111111001110
0111100111111111111111111111111111111111111110011110
0111110011111111111111111111111111111111111100111110
0111111001111111111111111111111111111111111001111110
0111111100111111111111111111111111111111110011111110
0111111110011111111111111111111111111111100111111110
0111111111001111111111111111111111111111001111111110
0111111111100111111111111111111111111110011111111110
0111111111110011111111111111111111111100111111111110
0111111111111001111111111111111111111001111111111110
0111111111111100111111111111111111110011111111111110
0111111111111110011111111111111111100111111111111110
0111111111111111001111111111111111001111111111111110
0111111111111111100111111111111100111111111111111110
0111111111111111110011111111111001111111111111111110
0111111111111111111001111111110011111111111111111110
0111111111111111111100111111100111111111111111111110
0111111111101111111110011001111111111011111111111110
0111111111111111111110011100111111111111111111111110
0111111111111111111100111111001111111111111111111110
0111111111111111111001111111100111111111111111111110
0111111111111111110011111111110011111111111111111110
0111111111111111100111111111111001111111111111111110
0111111111111111001111111111111100111111111111111110
0111111111111110011111111111111110011111111111111110
0111111111111100111111111111111111001111111111111110
0111111111111001111111111111111111100111111111111110
0111111111110011111111111111111111110011111111111110
0111111111100111111111111111111111111001111111111110
0111111111001111111111111111111111111100111111111110
0111111110011111111111111111111111111110011111111110
0111111100111111111111111111111111111111001111111110
0111111001111111111111111111111111111111100111111110
0111110011111111111111111111111111111111110011111110
0111100111111111111111111111111111111111111001111110
0111001111111111111111111111111111111111111100111110
0110011111111111111111111111111111111111111110010110
0100111111111111111111111111111111111111111110010010
0001111111111111111111111111111111111111111111000000
0011111111111111111111111111111111111111111111111100
0111111111111111111111111111111111111111111111111110
0000000000000000000000000000000000000000000000000000
```

your program when run as python3 polygons.py --file polys_2.txt should output

```
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:
```

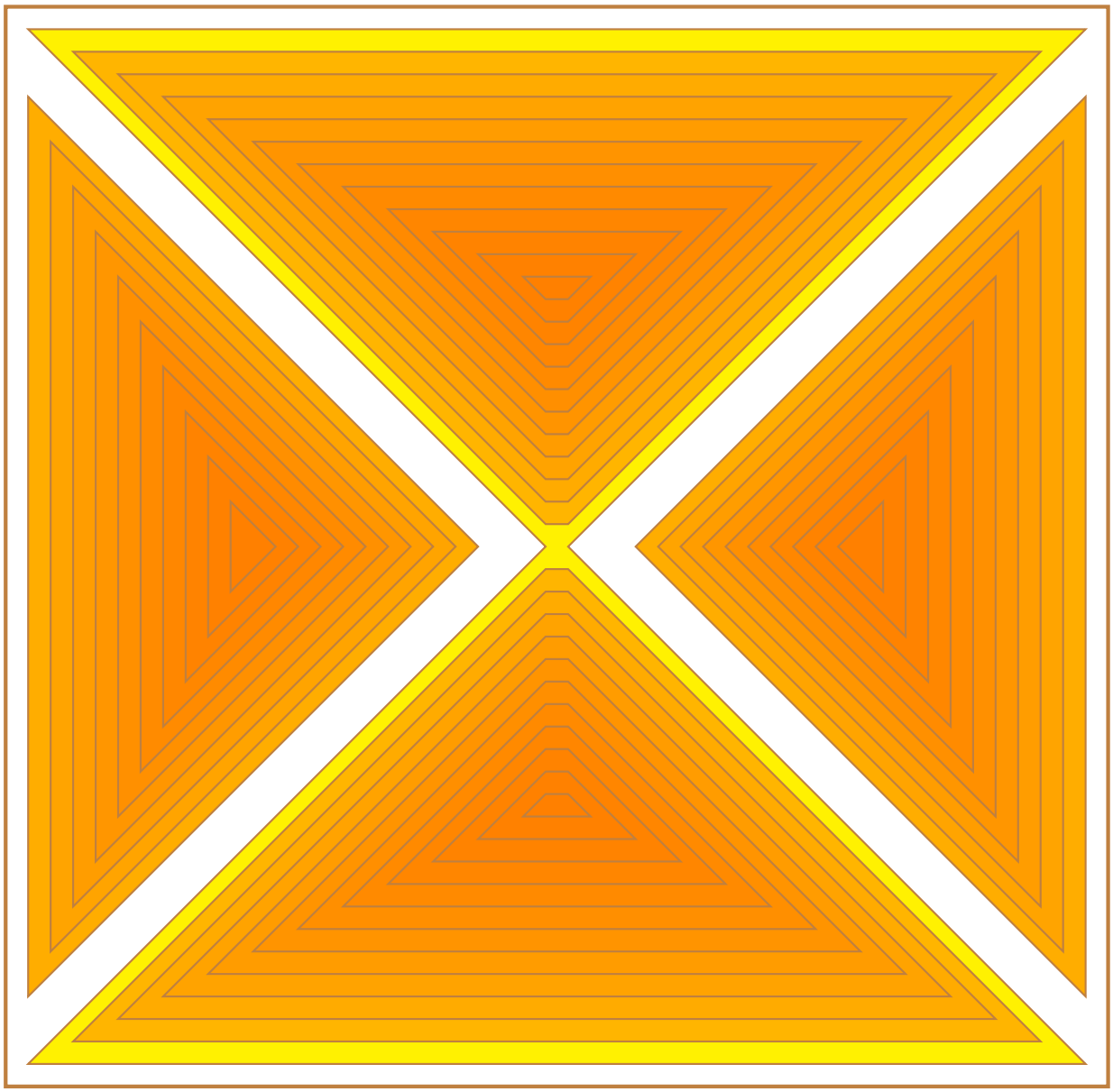
```
    Perimeter: 14.4 + 34*sqrt(.32)
    Area: 48.96
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 3
Polygon 6:
    Perimeter: 16.0 + 40*sqrt(.32)
    Area: 64.00
    Convex: yes
    Nb of invariant rotations: 1
    Depth: 0
Polygon 7:
    Perimeter: 12.8 + 30*sqrt(.32)
    Area: 38.40
    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
```

and when run as `python3 polygons.py -print --file polys_2.txt` should produce some output saved in a file named `polys_2.tex`, which can be given as argument to `pdflatex` to produce a file named `polys_2.pdf` that views as follows.



2.3 Third example

Given a file named `polys_3.txt` whose contents is

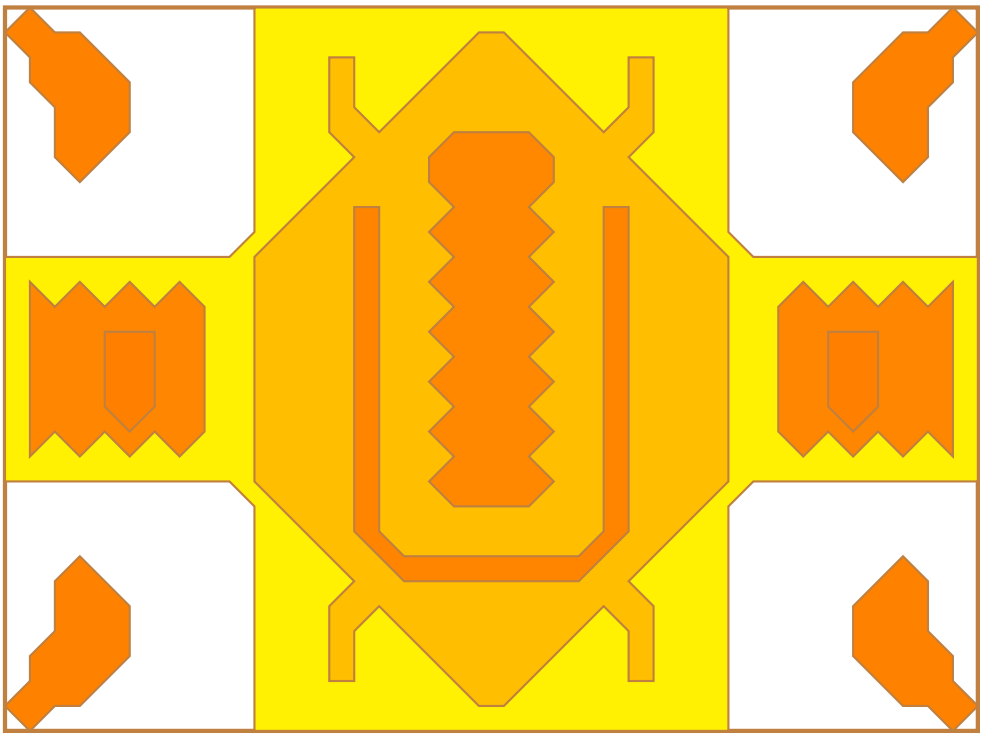
```
0 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 0
1 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 1
0 1 0 0 1 0 0 0 0 0 1 0 0 1 1 0 0 0 1 0 0 1 0 0 0 1 1 0 0 1 0 0 0 0 0 1 0 0 1 0
0 1 0 0 0 1 0 0 0 0 1 0 0 1 1 0 0 1 0 0 0 0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

your program when run as `python3 polygons.py --file polys_3.txt` should output

```
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
```

and when run as `python3 polygons.py -print --file polys_3.txt` should produce some output saved in a file named `polys_3.tex`, which can be given as argument to `pdflatex` to produce a file named `polys_3.pdf` that views as follows.



2.4 Fourth example

Given a file named `polys_4.txt` whose contents is

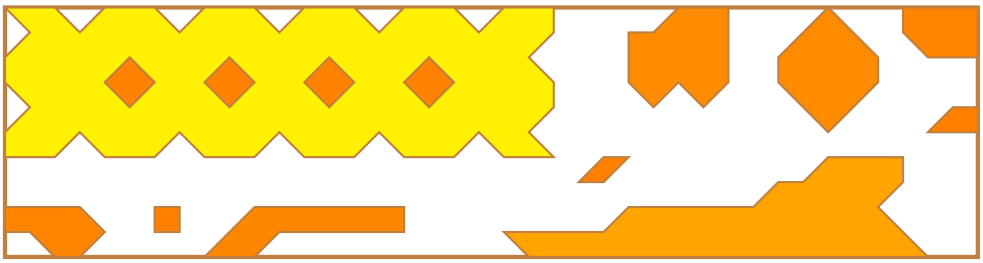
```
1    1  101     11 0  1 1            1 0 1      1 1011 10        1 1 1 0 000       1 1 1 0       00  1 001  11 1



01  01000100010001000100100                           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                 01110111011000000001111000
000000000000000000000001100000011000100                                                                  0

1                                  111001100111111100000000111111000      010000

110 01       0 1 1 0                                                      1011111100011111000000000001000

001  1000011       10          000000000                    1111111111111111                                 00
```

your program when run as `python3 polygons.py --file polys_4.txt` should output

```
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
```

and when run as `python3 polygons.py -print --file polys_4.txt` should produce some output saved in a file named `polys_4.tex`, which can be given as argument to `pdflatex` to produce a file named `polys_4.pdf` that views as follows.



3 Detailed description

3.1 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.

3.2 Output

The program should be run as either

`python3 polygons.py --file filename.txt`

or as

`python3 polygons.py -print --file filename.txt`

(where `filename.txt` is the name of a file that stores the input). You can study the program `ascii_art.py` from Lecture 7 to find out how this can be done.

If the input is incorrect, that is, does not satisfy the conditions spelled out in the previous section, then the program should print out a single line that reads

```
Incorrect input.
```

and immediately exit.

3.2.1 When the program is run without -print as command-line argument

If the input is correct, then the program should output a first line that reads one of

```
Cannot get polygons as expected.
```

in case 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.

Otherwise, the program should output a first line that reads

```
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. Note that your program should output no blank line. For a given test, the output of your program will be compared with the expected output; your program will pass the test if and only if both outputs are absolutely identical, character for character, including spaces. For the provided examples, the expected outputs are available in files that end in `_output.txt`. To check that the output of your program on those examples is correct, you can redirect it to a file and compare the contents of that file with the contents of the appropriate `_output.txt` file using the `diff` command. If `diff` silently exits then your program passes the test; otherwise it fails it. For instance, run

```
python3 polygons.py --file polys_1.txt >polys_1_my_output.txt
```

and then

```
diff polys_1_my_output.txt polys_1_output.txt
```

to check whether your program succeeds on the first provided example.

3.2.2 When the program is run with -print as command-line argument

If the input is correct, then the program should output some lines saved in a file named *filename*`.tex`, that can be given as an argument to `pdflatex` to produce a file named *filename*`.pdf` that depicts the maze. The provided examples will show you what *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 contents of the file output by your program 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. For instance, rename the provided file `polys_1.tex` to `polys_1_expected.tex`, and then run

```
python3 polygons.py -print --file polys_1.txt
```

and then

```
diff polys_1.tex polys_1_expected.tex
```

to check whether your program succeeds on the first provided example.