ORIGASMI
2013 ICTSA Programming Challenge

// Version 1.5

Christian Colombo, Jean-Paul Ebejer, Karl Fenech,
Gordon J. Pace, Chris Porter

November 27, 2013

This document outlines the programming challenge task that has to be solved in the 2013 ICTSA Programming Challenge. The competition opens on Friday, 29th Nov 2013 at 20:00 and closes on Sunday, 1st Dec 2013 at 20:00 ZST[1]. This document also specifies the criteria that will be used to judge the entries. The rules of the Programming Challenge can be found on the website `http://www.ictsamalta.org/?p=725`

1 Origasmi - paper folding joy

Countless authors, including J.K. Rowling and any student with a thesis submission deadline, have at some point in their careers suffered from writer's block. Indeed, in the words of William Goldman, '*The easiest thing to do on earth is not write*'. For more details on writer's block, please refer to Appendix A. A common remedy is procrastination, more specifically folding the paper you should be writing upon.

Your task this year is to write a computer program, which will automate this paper folding and enable prominent authors to get back to work.

1.1 A simple example

Origami, from "ori" meaning folding and "kami" meaning paper, is the traditional Japanese art of paper folding and is the inspiring theme behind this year's programming competition. The main difference is that in origami, a sheet of paper is folded into a three dimensional model (e.g. a crane) while in the programming competition you will be asked to approximate a two dimensional (flat) shape. Cutting the paper (e.g. with scissors) is not allowed.

[1]Zeus Standard Time — the time as set on UNIX server zeus, based at the Department of Computer Science at the University of Malta. This time can be seen on the Programming Challenge website

An example of how a paper may be folded to give a rather unceremonious two dimensional hat is shown in Figure 1.

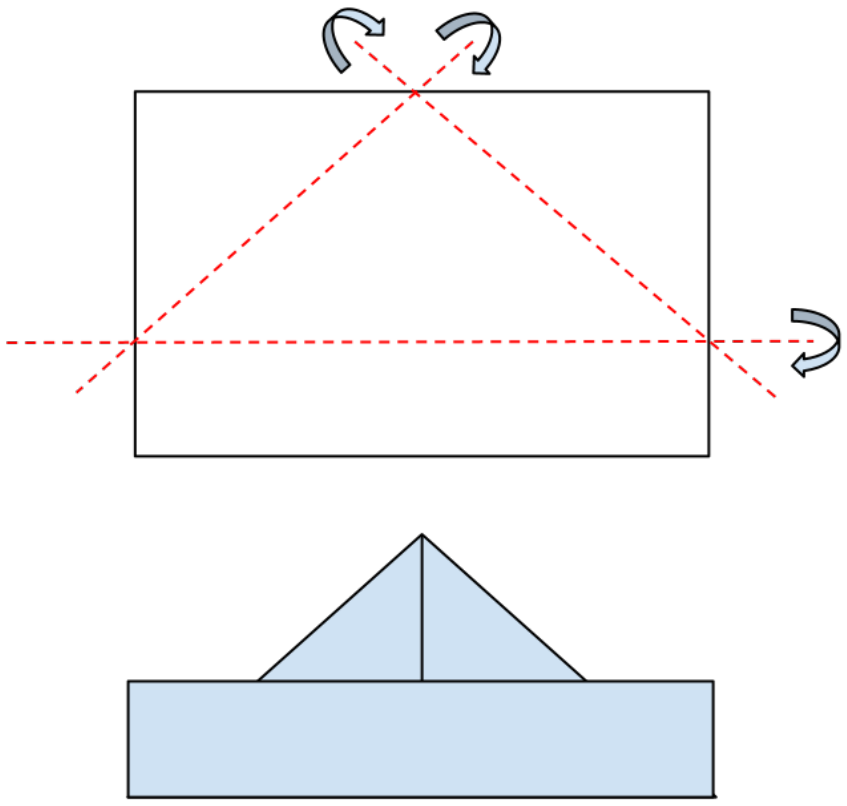


Figure 1: This rectangular paper folded at three different locations (shown with a red line), gives a two dimensional *kappell tal-bennej*.

1.2 Operators

Your code will be given a set of two dimensional shapes, which you need to approximate by applying three operations:-

$translate(dx, dy)$: Moves the paper dx units to the right, and dy upwards. Negative values are allowed (and effectively move the paper to the left and downwards).

$rotate(\theta)$: Rotates the sheet of paper around the origin by θ radians in a clockwise direction.

$fold()$: Folds the sheet of paper along the x-axis, placing the part below the x-axis above, and flattening the sheet completely before continuing.

1.3 Initial conditions – paper size, orientation and position

The paper size is of 2×1 units. Initially the paper is laid in landscape orientation with its lower left corner sitting at the origin. The co-ordinates of the four paper corners at the beginning are (0,0) for lower left, (2,0) for lower right, (2,1) for upper right, and (0,1) for upper left.

1.4 Scoring function

A fold is scored by placing a 2D grid over the paper. Equation (1) will be used to assess the quality of a paper fold.

$$score = \frac{max(|S \cap O| - |S \setminus O|, 0)}{|O|} \tag{1}$$

Where S is the set of grid points in the submitted shape; O is the set of grid points in the original (i.e. the shape you are trying to approximate). Note that this definition is governed by a two dimensional grid of points starting at (0,0). With an arbitrary chosen resolution of 0.015625 (1/64) on a 2×1 paper the grid consists of 129×65 points. Equation (1) therefore informally reads as the number of grid points in common between the submitted and the original shapes minus the number of points in the submission which are not in the original shape. We then divide this by the number of points in the original shape - this normalization factor allows us to compare differently sized puzzles. The *max* function guarantees that we always get a score in the interval [0.0, 1.0].

Note: Perfect scores (1.0) cannot be achieved for some particular shapes. (This may be for a number of reasons, including the 1,000 operations limit.) For these shapes, you should aim to come up with the best approximation possible.

2 The Task

2.1 Competition entry specification

We will run your entry using a command line similar to the below.

`origasmi` *shape_to_approximate_file output_op_file*

However your competition entry is run (see Section 4 for more details), it will take two command line arguments. These are described in the table below.

Command line argument	Description
shape_to_approixmate_file	The file containing the definition of the shape to approximate.
output_op_file	The output file where the origami folding operations will be written.

2.2 Shape file (`shape_to_approximate_file`)

This text file with extension `.shape` will contain a list of polygons, each of which will contain a list of (x, y) points. In the case of solid polygons (*i.e.* polygons without any "holes"), this file will contain only one polygon and the list of points will define the ordered vertices of the polygon. The last vertex defined for every polygon is connected to the first one to "close" the polygon. In the case of polygon with holes, *i.e.* an interior boundary, a number of polygons are specified. The union of these polygons specifies the final, complete shape which has to be approximated. Note that the `.shape` file may define two (or more) disjoint polygons, with different *polygon_id*s.

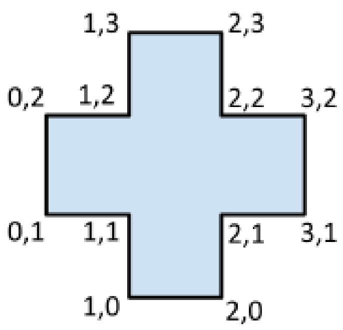
A `.shape` file is a text file, in which blank lines are ignored. Every line in this file is composed by a triplet of numbers representing a vertex of the polygon and has the following form:

```
polygon_id x y
```

Parameter	Domain	Description
polygon_id	$polygon\_id \in \mathbb{Z}$	The polygon identifier, used to determine when multiple polygons are specified (*i.e.* a polygon with a hole). The final shape is defined by the union of the polygons.
x	$x \in \mathbb{R}$	The x co-ordinate of a specific polygon vertex.
y	$y \in \mathbb{R}$	The y co-ordinate of a specific polygon vertex.

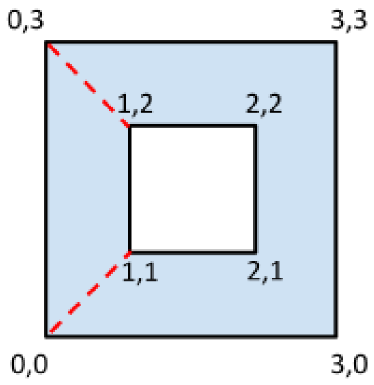
Examples of polygon shapes and their respective `.shape` files are shown in Table 1.

Table 1: Examples of two polygon specifying files



File: `plus.shape`

```
1 0 1
1 0 2
1 1 2
1 1 3
1 2 3
1 2 2
1 3 2
1 3 1
1 2 1
1 2 0
1 1 0
1 1 1
```



File: `hole_box.shape`

```
1 0 0
1 0 3
1 1 2
1 1 1
2 1 2
2 0 3
2 3 3
2 3 0
2 0 0
2 1 1
2 2 1
2 2 2
```

2.3 Output operations file (`output_op_file`)

Your program will, given a text file containing the shape to approximate, generate an origami operations file. This is a text file, with extension `.op`, which contains a single *origasmi* operation on every line. Blank lines will be ignored. Contrary to good software engineering principles, comments are **not** allowed in this file. The syntax described below is case sensitive, and space aware. The order of the operations in the file will define the order that these operations will be applied to the folding paper. **There is a 1,000 instructions limit to this file.** The syntax for the allowed operations is as follows:

2.3.1 Translate

```
T x y
```

Parameter	Domain	Description
x	$x \in \mathbb{R}$	The amount to move (translate) the paper horizontally. Positive values move the paper right, and negative values move the paper left.
y	$y \in \mathbb{R}$	The amount to move (translate) the paper vertically. Positive values move the paper up, and negative values move the paper down.

2.3.2 Rotate

```
R θ
```

Parameter	Domain	Description
θ	$\theta \in \mathbb{R}$	Angle in radians by which to rotate the paper. This rotation happens in a clockwise direction around the origin. Negative values will rotate the paper in an anti-clockwise motion.

2.3.3 Fold

The following command folds the paper along the x-axis.

```
F
```

2.3.4 Example .op file

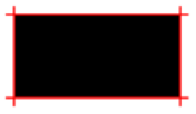
The following is an example of a working .op file. A step-by-step visual guide to these operations is given in Section 3.1.

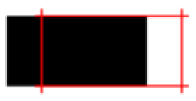
```
T -0.5 0.0
R 0.7853982
F
T -0.35355338 0.0
R 1.5707964
F
T 0.0 -0.70710677
F
T 0.0 -0.35355338
R 0.7853982
F
R 1.5707964
F
R 0.7853982
```

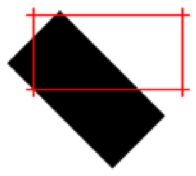
3 Worked Examples

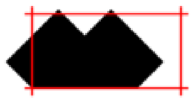
The following worked examples are available in the tools package (you may download these from the PC website http://www.cs.um.edu.mt/svrg/34t49ohheiufrgh45/Game_of_Codes/#task).

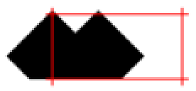
3.1 An arrow pointing west

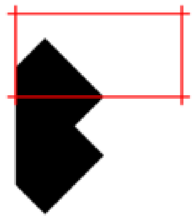

(a) Starting state

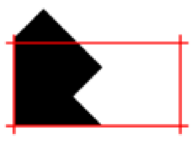

(b) `T -0.5 0.0`

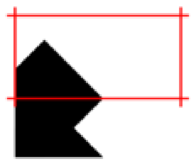

(c) `R 0.7853982`

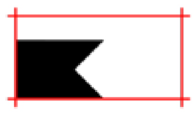

(d) `F`

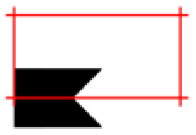

(e) `T -0.35355338 0.0`

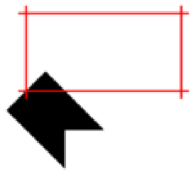

(f) `R 1.5707964`

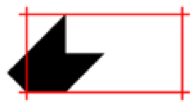

(g) `F`

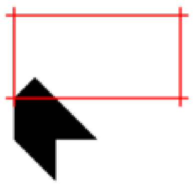

(h) `T 0.0 -0.70710677`

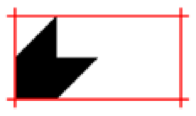

(i) `F`


(j) `T 0.0 -0.35355338`


(k) `R 0.7853982`


(l) `F`


(m) `R 1.5707964`


(n) `F`

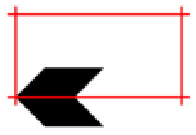
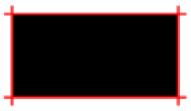
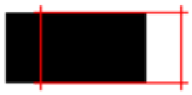
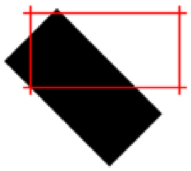

(o) `R 0.7853982`

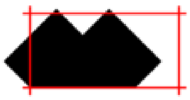
Figure 2: An arrow pointing west with the outline of the original paper shown in red.

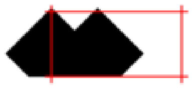
3.2 A pin hole needle

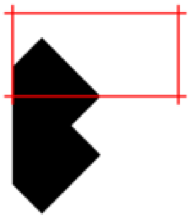

(3.1) Starting state

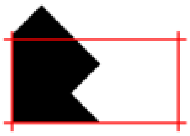

(3.2) `T -0.5 0.0`

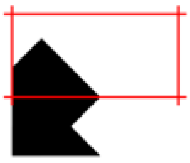

(3.3) `R 0.7853982`

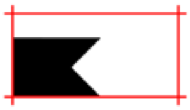

(3.4) `F`

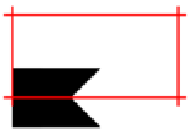

(3.5) `T -0.35355338 0.0`

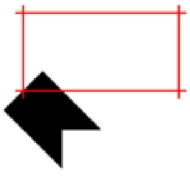

(3.6) `R 1.5707964`


(3.7) `F`


(3.8) `T 0.0 -0.70710677`


(3.9) `F`


(3.10) `T 0.0 -0.35355338`


(3.11) `R 0.7853982`

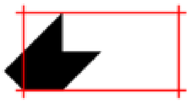
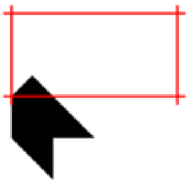
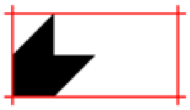
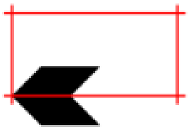
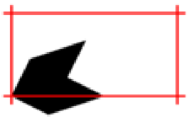

(3.12) `F`

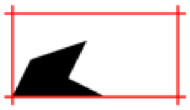
Figure 3: A pin-hole needle

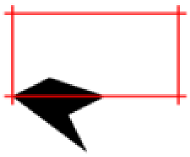

(3.13) `R 1.5707964`

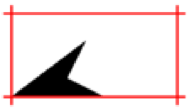

(3.14) `F`

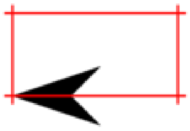

(3.15) `R 0.7853982`

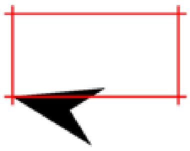

(3.16) `R -0.32175058`

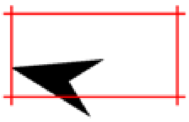

(3.17) `F`

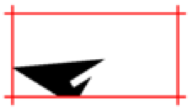

(3.18) `R 0.64350116`


(3.19) `F`


(3.20) `R 0.32175058`


(3.21) `R 0.23182383`


(3.22) `T 0.0 0.35`


(3.23) `F`

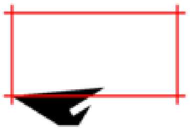
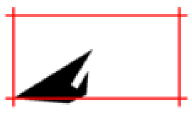
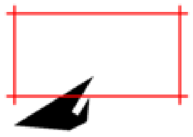
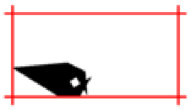
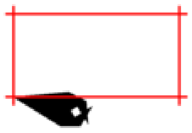

(3.24) `T 0.0 -0.35`

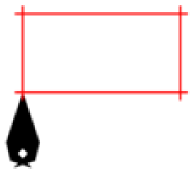
Figure 3: A pin-hole needle (continued)

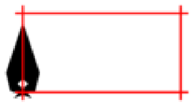

(3.25) `R -0.46364766`

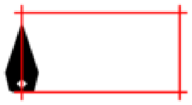

(3.26) `T 0.0 -0.35`


(3.27) `F`


(3.28) `T 0.0 -0.35`


(3.29) `R 1.3389726`


(3.30) `T 0.0 0.88`


(3.31) `F`

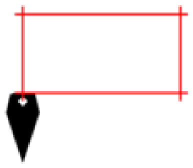

(3.32) `R 3.1415927`

Figure 3: A pin-hole needle (continued)

4 Submission Guidelines

A single competition entry program is to be called `origasmi`. You can submit an unlimited number of solutions throughout the weekend. The last submission on Sunday, 1st Dec 2013 at 20:00 ZST will be considered for the prize. After these times no corrections or resubmissions will be allowed. Submissions can only be done through the programming challenge website – note that the program must finish uploading before the aforementioned time.

There is no strict requirement about which programming language to use, but in the case of languages not mentioned in Table 2 an executable file which runs on either Ubuntu Linux 12.04.2 (64-bit) or Windows Server 2012 (64-bit) should be provided (you have to specify which).

Table 2: Supported competition programming languages and their versions.

Programming Language	Linux Version	Windows Version
Erlang	5.8.5	
Go	go1	1.1.2
Haskell	7.4.1	7.6.3
Java	1.7.0_25	1.7.0_45
Lisp	2.49	
Mono	2.10.8.1	
.NET		4.5
Perl	5.14.2	5.16.3
PHP	5.3.10	5.3.27
Python	2.7.3	2.7.6
Ruby	1.8.7	1.8.7

Depending on the programming language used there is some variation on how the competition entries will be run. In the case of Java, a jar file is to be supplied which will be run using `java -jar origasmi.jar`. In the case of interpreted languages (e.g. python, perl, ruby, php) the program will be run by calling the interpreter with the appropriate competition entry file. In the case of interpreted languages, your program should have the appropriate file extension (e.g. `python origasmi.py ...`). For compiled languages such as C or C++, the executable should be statically linked.

In any case the program has to accept the two command line arguments described earlier in Section 2.1.

No graphical user interfaces are to be opened by the program. The program will be executed on a AWS m1.large Instance with 4 EC2 Compute units cores and having 7.5GiB of RAM. The operating system will be either Ubuntu Linux 12.04.2 LTS (64-bit) or Windows Server 2012 (64-bit). For more details about Amazon AWS cloud instances please refer to `http://aws.amazon.com/ec2/#instance`. Any submissions that fail to execute on the mentioned environment or any submission that take longer than 1 minute to output a result will not be considered by the judging panel.

Also, there will be a special prize for instructions leading to the most original fold. This is a `.op` file submission, which is separate from the `origasmi` program.

5 In-competition support

During the period between Friday, 29th Nov 2013 at 20:00 and Sunday, 1st Dec 2013 at 20:00, all the teams can send queries for clarification through the use of the designated competition Google group which can be accessed by clicking the "Got a question?" button in the navigation menu of the website (`http://www.cs.um.edu.mt/svrg/34t49ohheiufrgh45/Game_of_Codes/`). No emails sent directly to the judges will be answered. If the judging panel deems the query to be a valid one, and not answered in this document, the answer will be posted back to the group. The judging panel will do its best to answer these queries in as short a time as possible.

6 Judging Panel & Criteria

The judging panel is made up by the authors of this document. Their decisions in any matters relating to the programming challenge are final and unappealable.

The judging panel will run all submissions on a number of `.shape` test cases. Each program will be given up to 1 minute to generate an operations file for each problem instance. Programs not terminating on a particular problem within 1 minute, or producing wrong output (*e.g.* illegal operations) will get no points for that problem. The `.op` file will be scored using a program which implements the scoring function described in Equation 1. We provide the scoring program along with this specification document.

For each shape, the programs will be ranked according to the highest score. A score is then given to each team according to their ranking: best 50 points, second 30 points, third 20 points, fourth 10 points and fifth 5 points. The rest will not be awarded any points.

In the case of ties on an individual problem, the team with the lowest number of operations is given priority. In case the tie is still not resolved, they will be given the same ranking for that problem.

The winning team will be the one which obtains the largest total number of points. In the unlikely case of unresolved ties, the ranking will be done according to the number of operations on all the problem set (lowest wins) after which the judging panel reserves the right to resolve any further tie in as fair a way as possible using a tie breaker.

Appendix A - Writer's Block Details