

Algorithms for Making Figures

(By using different kinds of primitives)

Instructions

- Given the following sets of *typed figures* (i.e., triangles and rectangles made up just with characters) design an algorithm/program to draw each figure on them, taking care of the constraints and tools stated in each question.
- Use a methodological approach to guide your work (we saw it the first two classes).
- Select two methods from the questions and write a **sound argument** (drawings at each step might be useful) to convince a classmate that your algorithm effectively produces figures
- Indicate what set of tools-primitives (instructions implemented in BASIC or Logo) you find better suited to solve the problems, and explain your reasons.

A	B	C	D	E
<i>N</i> -row cross	<i>N</i> -row triangle (rectangle)	<i>N</i> -row triangle (rectangle)	<i>N</i> -row triangle (isosceles)	<i>N</i> -row triangle (perimeter)
@ @ @ @ @ @ @ @ @	@ @@ @@@ @@@@ @@@@@	@@@@@ @@@@@ @@@@@ @@@@ @@@ @@	@@@@@@@@@@@ @@@@@@@@@ @@@@@@@@@ @@@@@@@@@ @@@@@	@@@@@@@@@@@ @ @ @ @ @ @ @
<i>N</i> = 5	<i>N</i> = 5	<i>N</i> = 5	<i>N</i> = 5	<i>N</i> = 5

F	G	H	I
<i>N</i> × <i>M</i> rectangle (solid)	<i>N</i> × <i>M</i> rectangle (perimeter)	4 <i>N</i> -row triangles (rectangles) (one after another)	Tree (pine)
@@@@@@@@@@@@@ @@@@@@@@@@@@@ @@@@@@@@@@@@@ @@@@@@@@@@@@@ @@@@@@@@@@@@@	@@@@@@@@@@@@@ @ @ @ @ @ @ @ @	@@@@@ @@@@@ @ @ @@@@ @@@@ @@ @@@ @@@ @@@ @@@ @@ @@ @@@@ @@@@ @ @ @@@@ @@@@	@ @@@ @@@@@ @@@@@@@@@ @@@ @@@
<i>N</i> = 5, <i>M</i> = 11	<i>N</i> = 5, <i>M</i> = 11	<i>N</i> = 5	<i>N</i> = 6, <i>Base</i> = <i>N</i> /3

Remember, the figures shown above are just examples, the number of rows (*N*) or columns (*M*) is variable, and the symbol “@” is just used as a reference to any character.

- Regard your workspace as a blank piece of white paper, and your tools are ruler and pencil. Instead of the character “@”, just make a dot-like mark “•”.
- Regard your workspace as a blank piece of graph paper, and your only tool is a pencil (the ruler is optional). Instead of the character “@”, just fill a square in with the pencil “■”.

3. Regard your workspace as a **XY-Cartesian system** and your tool is an algorithmic language (i.e. for numerical computations) with the command `print_dot (x,y)` that leaves a mark (“■”) in the XY-position specified, and also `print_blank (x,y)`, which leaves a blank space (“□”) in the XY-position specified.
4. Regard your workspace as a **polar coordinated system** with center (origin) is located at the center of your paper. Your tools are pencil, ruler, *angle measurer*, and an algorithmic language (i.e. for numerical computations) with the command `print_dot (r, α)` that leaves a mark (“•”) at a r distance from the center, with α degrees of elevation with respect to the horizontal. (**Note:** solid figures are not required in this exercise.)

You may find more information on polar coordinated systems at

<http://archives.math.utk.edu/visual.calculus/0/polar.6/> (simple)

http://www.gurusnetwork.com/tutorial/polar_coordinates/?PHPSESSID=4a69f3a754dfad54b96440863c77b291 (fun)

5. Regard your workspace as a **turtle graphics** system with center (*home*) located at the center of your paper. Your tools are pencil, ruler, *angle measurer*, and the programming language Logo. (**Note:** solid figures are not required in this exercise.)

You may find the information you need on Logo programming at

<http://el.media.mit.edu/logo-foundation/logo/turtle.html>

6. Regard your workspace as a blank piece of graph paper to represent **Karel's world**. Your tools are a pencil and the set of instructions recognized by the robot Karel. You may find the information you need on Logo programming at <http://home.att.net/~David.D.Barnett/tutorial.html>