Writing Snap! By Hand

Expressing and defining algorithms is a vital topic for this class. It absolutely pervades every area of computer science, and you need to be able to describe algorithms in a clear and unambiguous way - on paper as well as when you are working on a computer. The best language we have in this class, and the one that you have gained experience with, is Snap!. Unfortunately, as a graphical programming language, based around dragging out graphical puzzle pieces, it doesn't lend itself very well to writing on paper. Because of that, we have developed the following technique for sketching out Snap! code on paper.

When writing on paper, we will use simple visual annotations to mark the important parts of Snap! scripts and blocks. In particular, the important visual elements will be represented as shown below, where examples have the "hand drawn" version next to each Snap! example.

- Hat blocks: Designate by surrounding the event that triggers the hat block with lines. For example:

when green flag clicked

- Variables: When writing a variable, surround it with parentheses, so initializing a variable sCount to 0 would look like this:

set (sCount) to 0

- Block arguments: When you use a block, there are two distinct things that are shown: descriptive title words, and parameters. On the computer, the parameter slots show up as little "windows" that you can fill in. On paper, what we will do is underline parameters, like this:

go to x: 0 y: 0

Note that sometimes arguments can include blocks that also take parameters - in that case things would be underlined twice (or more!), as in the following example (note also the use of a variable):

Say join Hello (name) for 2 secs

- C-blocks: for any block that contains other blocks, draw a vertical line down from the top of the C-block that marks the blocks that it contains (and the blocks that are contained will be indented slightly to the right because of this):

repeat 8
move 10 steps

The if/else block has two "container" areas, so is a special case - it is drawn as follows:

if (value) mod 2 = 0
say Even
else
say Odd

Finally, as a bigger example, this is how we could write out a block definition for a block that adds up the values in a list:

sum of items in (pList)
script variables (sIndex) (sSum)
set (sIndex) to 1
set (sSum) to 0
repeat length of (pList)
set (sSum) to (sSum) + item (sIndex) of (pList)
change (sIndex) by 1
report (sSum)