Writing Snap! code on paper

You will be asked to write Snap! code on this exam, so we’ve developed a technique for writing it on paper. There are a few key things to notice:

- We often write variables in **UPPERCASE**.
- We change spaces between words in block names to dashes (this makes it much easier to read).
- We use indentation just as Snap! does, to help us understand what is “inside” the if, else, and other Control structures. E.g., here’s how you could write the DrawSquare and n! blocks:

```plaintext
DrawSquare(LENGTH)
  repeat(4)
    move(LENGTH) steps
turn-right(90) degrees

(n)!
  if n = 0
    report(1)
  report(n * (n - 1) !)
```

- When you want to write a list of things, write them with an open parenthesis, then the first item, second item, etc (separated by spaces) and when you’re done, put a closed parenthesis. If any of your items are a sentence, you have to put quotes around the sentence. So, for example, the following list of three things would be written as the equivalent 3-element list:

  ■ (life liberty "pursuit of happiness").

- Similarly, a nested list just shows up as a nested set of parenthesis. So the following would be written as

  ■ ((Love 5) (Hate 4) (The 10)).

- If you want to pass in a function as argument, you know the function must be surrounded by a grey-border. Here are three new conventions:

  ■ The grey border is written as *square brackets*: [ ]
  ■ Blanks are written as parenthesis with underscore _ in the middle, but common blocks that are passed in to HOFs can be simplified by just their name (and not the parens and underscores)
  ■ Return values are written as \( \rightarrow \) value

- So the following would be written as:

  ■ Map[ (\_)*(_) ]Reduce[ (_)+(_) ]over( (1 20 3 10) ) \( \rightarrow \) 510

- or, in the more simplified (and preferred) format:

  ■ Map[ * ]Reduce[ + ]over( (1 20 3 10) ) \( \rightarrow \) 510