Interview 1

This interview is concerned with **register machines**: a basic form of computation where instructions are represented as nodes and arrows in a graph.

No prior knowledge of register machines is expected.

Register machines contain a finite number of registers $R_0, R_1, \ldots, R_n$, each of which contains a non-negative integer. For example, a register machine could contain 3 registers R_0, R_1, R_2, which contain the following values: $R_0 = 5, R_1 = 0, R_2 = 7$.

There are four types of nodes in a register machine graph:

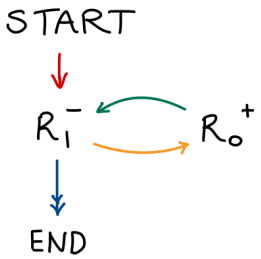
- START (only one START node)
- R_n^+ (where n refers to the register number)
- R_n^- (where n refers to the register number)
- END (at least one END node)

Computation starts at the START node, and ends at an END node. We update register values, and travel from one node to the next according to the following rules:

- START: Start here, and travel to the next node in the graph.
- R_n^+: Increment R_n by 1, and travel to the next node in the graph.
- R_n^-: If $R_n > 0$: decrement R_n by 1, and follow the single-tipped arrow.
  Else ($R_n = 0$): do not decrement R_n, and follow the double-tipped arrow.
- END: End computation

Register machines can be used to represent non-negative integer functions. The function's inputs are stored in $R_1, R_2, \ldots, R_n$, and the function's output is found in R_0 once an END node has been reached.

The following register machine represents the function $f(x) = x$. This function has one input, which is loaded into R_1 at the start of the program. At the end of the program (once we reach the END node) R_0 contains x, and so this is the output of the register machine.



The colours are not part of the register machine, and are solely included for explanation purposes.

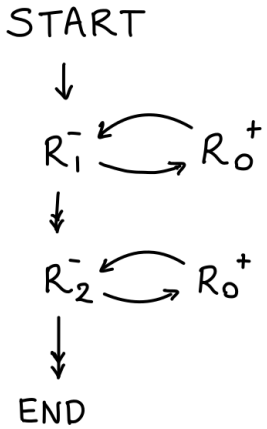
In this example we only use two registers, however you are free to introduce as many "dummy" registers as you would like (even if they do not correspond to function inputs or outputs). These registers, along with R_0, are initiated with the value 0.

If we were to execute the $f(x) = x$ register machine with input $x = 2$, the computation would look like:

1. Load the function argument ($x = 2$) into register R_1.
2. All remaining registers (in this case, just R_0) are initially 0.
3. Begin at the START instruction ($R_0 = 0, R_1 = 2$).
4. Travel from START to R_1^- along the red arrow.
5. Since $R_1 > 0$, decrement R_1 by 1 and follow the single-tipped yellow arrow ($R_0 = 0, R_1 = 1$).
6. Increment R_0 by 1, and travel to R_1^- along the green arrow ($R_0 = 1, R_1 = 1$).
7. Since $R_1 > 0$, decrement R_1 by 1 and follow the single-tipped yellow arrow ($R_0 = 1, R_1 = 0$).
8. Increment R_0 by 1, and travel to R_1^- along the green arrow ($R_0 = 2, R_1 = 0$).
9. Since $R_1 = 0$, do not decrement R_1 and follow the double-tipped blue arrow.
10. We have reached the END node, so terminate computation. The output of the function is the value in the R_0 register: 2.

Hence, if this register machine were to represent the function f, we have shown that $f(2) = 2$. Can you informally reason about why this machine represents the function $f(x) = x$ for all x?

What function does the following two-argument register machine represent?



Can you construct register machines for the following functions:

(1) $$f(x,y) = x \dot{-} y = \begin{cases} x - y & \text{if } x \geqslant y \\ 0 & \text{if } x < y \end{cases}$$

(2) $$f(x,y) = x \text{ div } y = \begin{cases} \text{the integer part of } x/y & \text{if } y > 0 \\ 0 & \text{if } y = 0 \end{cases}$$

(3) $$f(x) = 2^x$$

(4) $$f(x) = \begin{cases} \log_2 x & \text{if } x > 0 \\ 0 & \text{if } x = 0 \end{cases}$$

You are allowed to reuse register machines from previous questions. If you run out of time, try sketching a general outline of the register machine.