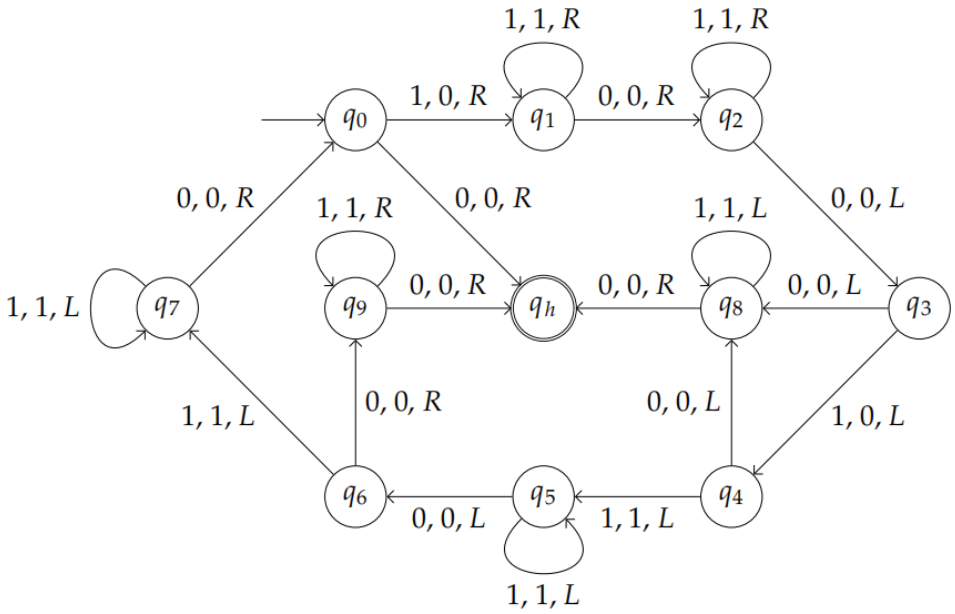
(1) Let a Turing machine have the following flowchart.



It is worth noting that we can interpret this machine as computing the "distance" function,

$$\mathrm{dist} : \mathbb{N}^2 \to \mathbb{N} :: (n, m) \mapsto |n - m| = \begin{cases} n - m & \text{if } n \geqslant m, \\ m - n & \text{otherwise.} \end{cases}$$

Now, given the following initial "complete configuration" (or "instantaneous descriptions"; see the *Turing Machines* chapter slide deck, p. 14) of the machine and the tape, figure out the successive configurations from the initial to the halting.

$$q_0 111011$$

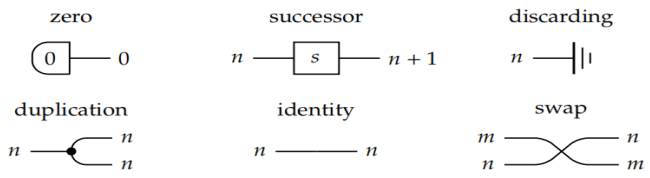
This notation assumes that, apart from these 1s, all the other squares have 0s. In your answer, keep 0s in the squares that the machine is scanning or has scanned at least once: e.g., if the machine in the above configuration moves to the right after rewriting 1 into 0, then you need to write $0q_i11$ as opposed to q_i11 for the next configuration, because the square with the 0 has been scanned.

(2) Design a Turing machine that computes the following function $\mathrm{IsZero} : \mathbb{N} \to \mathbb{N}$ and draw a flowchart for it.

$$f(n) = \begin{cases} n + 1 & \text{if } n \text{ is odd,} \\ n & \text{if } n \text{ is even.} \end{cases}$$

You can use any representation of natural numbers, but you need to explain how the representation of your choice works. Also, do not forget to specify where on the tape the machine should be placed initially.

(3) In our definition of primitive (or partial) recursive functions we use the box-and-wire notation and take the following six functions as basic (see pp. 2, 16, and 21 of the *Recursive Functions* chapter slide deck):



Using the box-and-wire notation and our definition, show that the function

$$f : \mathbb{N}^3 \to \mathbb{N}^3 :: (n, m, \ell) \mapsto (m, 2, m + 2)$$

is primitive recursive. You *cannot* use the fact that addition is primitive recursive.

(4) Prove that the following function,

$$\mathrm{IsZero} : \mathbb{N} \to \mathbb{N} :: n \mapsto \begin{cases} 1 & \text{if } n = 0, \\ 0 & \text{if } n > 0, \end{cases}$$

is primitive recursive, by the following steps:

(a) Define IsZero by primitive recursion. This is to spell out the two functions f and g used in the base clause (the 0th step, using f) and the inductive clause (the ith step within the `for` loop, using g) in primitive recursion defining $h = \mathrm{IsZero}$, as in

```
IsZero(0) := f
for i in (0, ..., n-1):
    IsZero(i+1) := g(i, IsZero(i))
```

Or, in even simpler words, find two functions f and g such that

$$\mathrm{IsZero}(0) = f, \qquad \mathrm{IsZero}(i+1) = g(i, \mathrm{pred}(i)).$$

(b) Show that these two functions f and g are both primitive recursive.

(5) Keep applying β-reduction to the following term until it no longer applies.

$$(((\lambda x.(\lambda y.((xy)\,y)))\,a)\,b) \xrightarrow{\beta} ? \xrightarrow{\beta} \cdots.$$

Show all the steps of β-reduction explicitly. We assume that neither x nor y occurs in a or b.

(6) In the (untyped) lambda calculus, "Church numerals" are defined in such a way that

$$\bar{3} := (\lambda x.(\lambda y.(x(x(xy))))), \qquad \bar{4} := (\lambda x.(\lambda y.(x(x(x(xy)))))).$$

(see p. 6 of the *Lambda Calculus* chapter slide deck). Let us also define

$$\mathrm{Succ} := (\lambda u.(\lambda x.(\lambda y.(x((ux)\,y))))).$$

Then prove either that

$$(\mathrm{Succ}\,\bar{3}) \xrightarrow{\beta} \cdots \xrightarrow{\beta} \bar{4}$$

or equivalently (but replacing x and y in $\bar{3}$ with v and w) that

$$(\mathrm{Succ}\,(\lambda v.(\lambda w.(v(v(vw)))))) \xrightarrow{\beta} \cdots \xrightarrow{\beta} \bar{4},$$

by showing all the steps of β-reduction explicitly.

(b) Show that the condition you have filled in in (a) actually holds.

(8) For each of the following (a) – (f), answer whether it is true or false.

(a) Every partial recursive function is primitive recursive.
(b) Given any Turing machine, the function it computes is partial recursive.
(c) For every term in the lambda calculus, there is a way to keep applying β-reduction to it and reduce it to a term to which β-reduction no longer applies.
(d) Every λ-definable function is representable in arithmetic.
(e) The Church-Turing thesis was first put forward by Church and then mathematically proven by Turing.
(f) The Church-Turing thesis implies that any algorithm can be performed by some Turing machine.

Part II. *A bit of application to computability theory.* In this part, by "computable" we mean any of the equivalent conditions "Turing computable", "partial recursive", and "λ-definable".

(7) In class we used, without proving, the fact that the relation between a pair of numbers (n, m), "$n = m$", is decidable. Prove this fact, by the following steps:

(a) Using the definition of decidability (p. 8 of the *Computability* chapter slide deck) in terms of computable functions (rather than algorithms), state what it means for "$n = m$" to be decidable, i.e., fill in the blank in

"$n = m$" is decidable if and only if ....

(b) Show that the condition you have filled in in (a) actually holds.