Theoretical Computer Science

1. Knights and Knaves

You wake up. Wait, where are you? You groggily come to your senses. The last thing you remember was falling asleep during a discrete math final. You look around, and spot a helpful sign:

> THE ISLAND OF KNIGHTS AND KNAVES
>
> Welcome to our cozy isle! All our inhabitants are very friendly, but be warned: half of them are knaves, who always lie. The other half, happily, are knights, who always tell the truth. Have a nice time!

Maybe if you wander around a bit, you can figure out how to get back home…

Please justify all your answers for this section.

1.1 The first encounter

You meet two inhabitants, Rémi and Miguel. Rémi says something to you, but he mumbles and you're unable to make it out. Miguel notices that you didn't catch it, and says "Rémi said that he is a knave." Is Rémi a knight or a knave? Is Miguel a knight or a knave?

1.2 The second encounter

You meet another inhabitant, who says "I am about to tell you the truth about whether I'm a knight or a knave." He makes another statement, and true to his word, you then know with 100% certainty whether he's a knight or a knave. What was his second statement, and is he a knight or a knave?

1.3 A conundrum

Is it possible for any inhabitant to say "You can't deduce if I'm a knight or a knave from this statement"?

2. SKI Combinator Calculus

In this system, there are 3 rules: S, K, and I.

I is the identity combinator: the pattern Ix for any x is replaced with x.
K is constant combinator. The pattern Kxy for any x and y is replaced with x.
S follows the rule $Sxyz = xz(yz)$, for any x, y, and z.[1]

The leftmost pattern is always reduced, and expressions can be grouped with brackets. For example, $IKIS$ is the same as $(IK)IS$ because IK is the leftmost pattern. We can reduce the IK inside the brackets to K, leaving us with KIS. The, by K's rule, this leaves us with I.

As another example, consider the combinator $Mx = xx$, which replaces something with two copies of itself. One way to define it using S, K, and I is like this: $M = SII$. Then, $Mx = SIIx = Ix(Ix) = x(Ix) = xx$, as intended. This can no longer be reduced further, as x is an arbitrary variable, and its reduction rule is unknown. Perhaps x is S, perhaps it is K, perhaps it is something else entirely.

Note that $x(Ix)$ is *not* the same as xIx. In fact, we can't reduce xIx at all, because once again x is a variable and we don't know what it does.

More examples:
- $Kx(Sz)$ reduces to x
- $K(Sxyz)I$ reduces to $xz(yz)$
- KS reduces to KS (we don't have enough info to continue reduction, so we leave it)

2.1 False

One way to represent a boolean false is with a function that discards its first argument: $Fxy = y$. Define this in terms of S, K, and I.

2.2 Idiot

It turns out that you don't actually need I! Define it in terms of S and K. (I is sometimes known as the *idiot bird*.)

2.3 Compose

Define the B combinator, which follows the rule $Bxyz = x(yz)$. In other words, it composes x and y.

---

[1] S's rule may seem arbitrarily, but it encodes the very common pattern of calling a function with some input and the same input but modified. For example, a string if is a palindrome if it's equal to its reverse. Here, the function is equality, the string is the input, and the modified input is the reversed string.

```
def eq(x, y): return x == y
def reverse(s): return s[::-1]
def S(x, y, z): return x(z, y(z))
def palindrome(s): return S(eq, reverse, s)

print(palindrome("racecar"))  # True
print(palindrome("starling"))  # False
print(palindrome(re.sub("\\W","","A man, a plan, a canal - Panama!").lower()))  # True
```