**Swhich** is similar to **Eel** and **Millipede**, but it uses two "keystrings" instead of one, and both "roll into" the other, as if they were twisted, hence the name.

```
function swhich!(f)
    f[1] = [f[1][1:end];f[2][1]]
    f[2] = [f[2][2:end];f[1][1]]
    popfirst!(f[1])
end
```

In the encoding function, the leading entries of both keystrings are added mod $n$ to the plaintext symbol to get the ciphertext symbol. Then the "tapes" (keystrings) are tangled/swhiched. Finally the new top-left entry is modified. The plaintext symbol is added to it (again, mod $n$). This means the plaintext symbol affects the key immediately before it acts on the *next* plaintext symbol. (Note that Julia is 1-indexed.)

```
function encode(p,q,n)
    f = deepcopy(q)
    c = Int64[]
    for i in eachindex(p)
        push!(c, ( f[1][1] + f[2][1] + p[i] )%n )
        swhich!(f)
        f[1][1] = (f[1][1] + p[i])%n
    end
    c
end
```

To move from encoding to encryption, we do the usual reversal between rounds. We also "autospin" and make the length of key the default number of rounds. In this case the autospin is a simple circular shift.

```
function encrypt(p,q,n)
    l = length(q)
    for i in 1:l
        f = deepcopy(q)
        f = circshift(f,i)
        p = encode(p,f,n)
        p = reverse(p)
    end
    p
end
```