CS302: Paradigms of Programming

Lab 3: States and Mutation

March 30^{th}, 2021

Q1. Modify the `make-account` procedure (from the class) so that it creates password-protected accounts. That is, `make-account` should take a symbol as an additional argument, as in:

```
> (define acc (make-account 100 'secret-password))
```

The resulting account object should process a request only if it is accompanied by the password with which the account was created, and should otherwise return a complaint:

```
> ((acc 'secret-password 'withdraw) 40)
> 60
> ((acc 'nakli-password 'withdraw) 50)
> "Incorrect password"
```

Q2. Modify the `make-account` procedure of the previous question by adding another local state variable, such that if an account is accessed more than five consecutive times with an incorrect password, it invokes the procedure `call-the-cops` (use your creativity to implement this final procedure).

Q3. In software-testing applications, it is useful to be able to count the number of times a given procedure is called during the course of a computation (a kind of 'profiling'). Write a procedure `make-monitored` that takes as input a procedure, `f`, that itself takes one input. The result returned by `make-monitored` is a third procedure, say `mf`, that keeps track of the number of times it has been called by maintaining an internal counter. If the input to `mf` is the special symbol `'how-many-calls?`, then `mf` returns the value of the counter. If the input is the special symbol `'reset-count`, then `mf` resets the counter to zero. For any other input, `mf` returns the result of calling `f` on that input and increments the counter. For instance, we could make a monitored version of the `square` procedure:

```
> (define s (make-monitored square))
> (s 100)
> 10
> (s 'how-many-calls?)
> 1
```