Homework 1: DSSL2 Warmup

The purpose of this assignment is to get you programming fluently in DSSL2, the language that we'll be using for the course.

On Canvas, you will find starter code (`warmup.rkt`) including some definitions, headers for the methods and functions that you'll need to write, along with an insufficient number of tests.

Part I: Installing DSSL2

To complete this homework assignment, you will first need to install the DrRacket programming environment, version 8.0 (available from `racket-lang.org`). Then you will need to install the DSSL2 language within DrRacket.

Once you have DrRacket installed, open it and choose "Package Manager" from the "File" menu. Type `dssl2` as the source, then click the "Install" button and wait for installation to finish. When it's finished, the "Install" button should change to "Update"; then close the window.

Part II: Class practice

In `warmup.rkt` we provide a partial definition for the class `Account`, which represents bank accounts. Accounts include

- an `id`entifier, which is a natural number that uniquely identifies the account,
- an account `type`, which can be either `"checking"` or `"savings"`,
- and a `balance`, which must be non-negative.

We also provided the constructor, which checks the above constraints of `type`s and initial `balance`s, as well as getter methods for the fields.

The `Account` class also contains headers for three additional methods, which you must write:

1. `Account.deposit(num?) -> None`, which adds an amount to the balance of the account. The amount deposited must be non-negative.
2. `Account.withdraw(num?) -> None`, which subtracts the given amount from the balance. If the requested withdrawal exceeds the balance, this method must call `error` instead. The amount withdrawn must be non-negative.
3. `Account.__eq__(Account?) -> bool?` compares two accounts for equality by comparing their three fields. Two accounts are equal if all three fields are equal, pairwise.

Note that the `__eq__` method implements the `==` operator for its class. That is, comparing two `Account`s with `==` calls your `__eq__` method, which must be defined to compare objects of its class appropriately.

Then, there is one free (non-method) function dealing with accounts for you to write:

4. `account_transfer(num?, Account?, Account?) -> None` withdraws the given number from the first account's balance and deposits it in the second. The amount transferred must be non-negative, and must not exceed the first account's balance.

You will want to write additional tests for the above methods and function.

Part III: Customers

The next set of functions you will write work with vectors of `customer` structs. Customer structs have two fields, `name` which should be a string, and `bank_account` which should be an `Account` built using the `Account` class described above. Multiple `customer`s can have the same name.

5. `max_account_id(VecC[customer?]) -> nat?` takes a vector of customers, and returns the highest account `id` found in any of the given customers' accounts. This function must raise an error when no customers are given.
6. `open_account(str?, account_type?, VecC[customer?])`
   `-> VecC[customer?]` takes the name of a customer, an account type, and a vector of customers, and produces a new vector of customers with a new customer added. That new customer must have the provided name, and have a newly created account with the given type, a balance of 0, and an id one higher than the highest previously-used id. If this new account would be the first account in the ledger, start with an id of 1.
7. `check_sharing(VecC[customer?]) -> bool?` takes a vector of customers, and checks whether any of the customers in the vector have accounts with identical `id`. If that's the case, return `True`, otherwise return `False`.

As before, you will want to write additional tests for the above operations.

Part IV: Honor code

Every homework assignment you hand in must begin with the following definition (taken from the Provost's website[1]; see that for a more detailed explanation of these points):

```
let eight_principles = ["Know your rights.",
     "Acknowledge your sources.",
     "Protect your work.",
     "Avoid suspicion.",
     "Do your own work.",
     "Never falsify a record or permit another person to do so.",
     "Never fabricate data, citations, or experimental results.",
     "Always tell the truth when discussing your work with your instructor."]
```

If the definition is not present, you receive no credit for the assignment.

Note: Be careful about formatting the above in your source code! Depending on your pdf reader, directly copy-pasting may not yield valid DSSL2 formatting. To avoid surprises, be sure to test your code *after* copying the above definition.

Deliverable

Submit the provided file `warmup.rkt`, containing definitions for:

- the methods and functions described above,
- sufficient tests to be confident of your code's correctness,
- and the honor code.

Your code will be evaluated for correctness, efficiency, style, and thoroughness. In the future you will be graded for resource efficiency as well.

Submission

Your homework must be submitted via Canvas.

[1] `http://www.northwestern.edu/provost/students/integrity/rules.html`