CS51 Lab 19 Instructions

Reminders

- In today's lab, you'll be working in groups of four.
- Today's lab is a synthesis lab. No new material, but much more latitude in how you and your group solve the lab.
- While we're getting underway, go ahead and download the lab materials as per the setup instructions.
- You can register your attendance at http://url.cs51.io/labcode with today's code, which will be distributed during the breakout session.
- Do you have hints about the course? Let me know: shieber@seas.harvard.edu.

The usual setup instructions

1. Head to http://url.cs51.io/lab19 and follow the instructions there to create your lab repository.
2. In the terminal, clone the repository using the shell command

   ```
   % git clone git@github.com:cs51/lab19-⟨yourname⟩.git lab19
   ```

   or

   ```
   % git clone https://github.com/cs51/lab19-⟨yourname⟩.git lab19
   ```

3. Go to the directory for your just-created local repository:

   ```
   % cd lab19
   ```

4. Open up the file `lab19.ml` in your favorite text editor.

   You're now ready to work on the lab.

An ATM Emulator

This lab is quite different in format from previous labs (as is the next one). We're giving you much more latitude in the design of your code, so that you can deploy whatever techniques from the course you see fit. Everyone at your table will be working together as a single group.

You'll see in the file `lab19.ml` that we've provided an implementation of a simple ATM emulator. Running our solution version of the emulator generates the following transcript.

```
% ./lab19.byte
Enter customer id: 314159
Welcome Emily
Enter action: (B) Balance (-) Withdraw (+) Deposit (=) Done (X) Exit: B
Current balance is: 100
Enter action: (B) Balance (-) Withdraw (+) Deposit (=) Done (X) Exit: +
Enter amount: 150
Current balance is: 250
Enter action: (B) Balance (-) Withdraw (+) Deposit (=) Done (X) Exit: -
Enter amount: 80
Here's your cash: [20 @ 20][20 @ 20][20 @ 20][20 @ 20] and 0 more
Current balance is: 170
Enter action: (B) Balance (-) Withdraw (+) Deposit (=) Done (X) Exit: =
So long.

Enter customer id: 45
Invalid id
Enter customer id: asdfghjkl
Invalid id
Enter customer id: 141421
Welcome Jacob
Enter action: (B) Balance (-) Withdraw (+) Deposit (=) Done (X) Exit: -
Enter amount: 53
Insufficient funds
Current balance is: 0
Enter action: (B) Balance (-) Withdraw (+) Deposit (=) Done (X) Exit: +
Enter amount: 300
Current balance is: 300
Enter action: (B) Balance (-) Withdraw (+) Deposit (=) Done (X) Exit: -
Enter amount: 53
Here's your cash: [20 @ 20][20 @ 20] and 13 more
Current balance is: 247
Enter action: (B) Balance (-) Withdraw (+) Deposit (=) Done (X) Exit: X
Exiting the ATM emulation
```

The ATM emulator is implemented as the function `atm`, which makes use of a set of component functions in the `ATMcomponents` module. These are functions that implement the primitive actions of the ATM machine: prompting for and acquiring information from the customer; querying and updating an account database; presenting information to the customer or dispensing cash. We've provided the module signature in `aTMcomponents.mli` but *you'll need to provide its implementation in a file `aTMcomponents.ml`*. (Hint: for acquiring information from `stdin`, you may want to use functions from the `Scanf` module or functions `read_line` or `read_int` from `Stdlib`.)

Your implementation will of course require some way of tracking all of the accounts, each with a customer name and a balance. You should assign two members of your table group to implement code for tracking accounts, which can be used by the other two members who are implementing the ATM components. You'll want to agree on an interface between the two pairs of implementers. How you split up the responsibilities, design the interface, and implement the parts is entirely up to you.

You'll submit your lab to Gradescope as a single group submission. Designate one person at your table as the lab submitter, who should submit the lab on behalf of the table, making sure to add all the members of your group to the submission so all will get credit for the lab. (Because of Gradescope limitations, the members of the group will need to be re-added every time the designated submitter submits.) We'll run some limited unit tests against your `ATMcomponents` implementation, so please don't modify the `aTMcomponents.mli` file. We won't be able to test any other code you write, except as it affects the behavior of `ATMcomponents`. You're on your own for that.