CS51 Lab 7 Instructions

Getting started

While students are coming into class, feel free to go ahead and download the lab materials in preparation for the start of the lab proper. (But please wait until you're asked before beginning work on the lab.)

Please leave this handout on the table for the next lab session.

New download procedure

1. Head to http://url.cs51.io/lab7 or https://classroom.github.com/a/G6EMIblQ and follow the instructions there to create your lab repository. (If neither works and a TF advises, fork and then clone the repository at http://github.com/cs51/lab7 and proceed as usual.) *Do not click the button to create a README.*

2. In the terminal, clone the repository using the shell command

```
% git clone https://github.com/cs51/lab7-⟨yourname⟩.git lab7
```

3. You'll get an empty repository warning. *This is expected!*

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

```
% cd lab7
```

5. In the directory for your just-created local repository, execute the shell command

```
% git remote add distribution https://github.com/cs51/lab7.git
```

   to associate your local repository with our distribution repository.

6. Execute the shell command

```
% git pull distribution master
```

   to pull all of the lab files (from the master branch of our distribution repository) into your local repository.

7. You should now have all of the files in your local repository. Execute the shell command

```
% git commit -am "initial distribution"

% git push
```

   which will stage, commit, and push the files to your personal remote repository.

A modular puzzle

Consider the following signature for an integer queue module and its corresponding implementation, as presented in the textbook.

```
module type INT_QUEUE =
  sig
    type int_queue
    val empty_queue : int_queue
    val enqueue : int -> int_queue -> int_queue
    val dequeue : int_queue -> int * int_queue
  end ;;

module IntQueue : INT_QUEUE =
  struct
    type int_queue = int list
    let empty_queue : int_queue = []
    let enqueue (elt : int) (q : int_queue) : int_queue =
      q @ [elt]
    let dequeue (q : int_queue) : int * int_queue =
      match q with
      | [] -> raise (Invalid_argument "dequeue: empty queue")
      | hd :: tl -> hd, tl
  end ;;
```

Which of the following expressions are well formed and well typed? Check all that apply.

1. `INT_QUEUE.empty_queue ;;`

2. `IntQueue.enqueue 1 [] ;;`

3.
```
let open IntQueue in
empty_queue ;;
```

4. `fun (q : INT_QUEUE) -> INT_QUEUE.enqueue 1 q ;;`

5. `fun (q : IntQueue) -> IntQueue.enqueue 1 q ;;`

6. `fun (q : int_queue) -> IntQueue.enqueue 1 q ;;`

7. `fun (q : IntQueue.int_queue) -> IntQueue.enqueue 1 q ;;`