CS51 Second Midterm Examination Solutions, Spring 2021

Q1:

Box should be checked

Q2.1:

```
let nat_gen : unit -> int =
  let current = ref ~-1 in
  fun () -> incr current;
            !current ;;
```

Q2.2:

The pure language obeys Leibniz's law that *two instances of the same expression in the same context evaluate to the same value*. But two instances of `nats_gen ()` in the same (empty) context evaluate to two different values. Thus `nats_gen` cannot be implemented in the pure language.

Q2.3:

```
let square_gen : unit -> int =
  let current = ref 0 in
  fun () -> incr current;
        !current * !current ;;
```

Q2.4:

```
let rec stream_of_gen (gen : unit -> 'a) : 'a stream =
  lazy (Cons (gen (), stream_of_gen gen)) ;;
```

Q2.5:

```
let gen_of_list (lst : 'a list) : unit -> 'a =
  let stored = ref lst in
  fun () -> match !stored with
        | [] -> raise NoMore
        | first :: rest ->
           stored := rest;
           first ;;
```

Q3.1:

1. x, y
2. none
3. f
4. x
5. none

Q3.2:

1. `fun y -> 42`
2. `(fun y -> 42) 21`
3. `let x = 42 + 1 in x + 2`
4. `let z = 5 in f (x + 1)`

Q3.3:

(e) `3`

(f) `{l1 -> 3}`

(i) `42`

(j) `{l1 -> 3}`

(k) `()`

(l) `{l1 -> 42}`

Q3.4:

```
let x = ref 3 in x := !x * 14
```

Q4.1:

Big-O provides an upper bound on time only for inputs larger than some threshold n_0. Presumably the thresholds for the two algorithms are such that the Karatsuba algorithm is faster only beyond 70 bignum digits in Python.

Q4.2:

This sorting algorithm has a time complexity of O(n^2), so that it falls within every larger complexity class as well. Thus the following boxes should be checked: 2, 4, 5, 6, 8, 9, and 11.

Q5.1:

```
class raichu : pokemon =
  object
    inherit pokemon "Raichu" 122
  end
```

Q5.2:

```
class pikachu : pokemon =
  object (this)
    inherit pokemon "Pikachu" 82 as super
    method! evolve =
      super#evolve;
      this#set_successor (Some (new raichu))
  end
```