Second Midterm Exam

STUDENT NAME

Search students by name or email...

Q1 Instructions and affirmation

0 Points

Before beginning this exam, make sure you have read and understood **the exam instructions** that were distributed ahead of the exam and are available at **https://url.cs51.io/exam-instructions**, and check the box below.

- [ ] I affirm my awareness of the standards of the Harvard College Honor Code.

Save Answer

Q2 Generators

23 Points

A *generator* for a sequence is a function of type `unit -> 'a` for some type `'a` that returns the next element in the sequence each time it is called. For instance, here is a (trivial) generator for the "constant" sequence `1, 1, 1, ...`.

```
let one_gen () = 1 ;;
```

Generators are a common and quite useful mechanism, featured prominently, for instance, in the Python programming language

prominently, for instance, in the Python programming language.

Q2.1

5 Points

The constant generator above is boring; it generates the same value each time it is called. Generators are interesting and useful in particular because they can generate *different* values.

Define a generator `nats_gen` that generates the natural numbers as `int`s. That is, on successive calls, it yields successive natural numbers, like this:

```
# nats_gen () ;;
- : int = 0
# nats_gen () ;;
- : int = 1
# nats_gen () ;;
- : int = 2
# nats_gen () ;;
- : int = 3
```

Enter your answer here

Save Answer

Q2.2

3 Points

Can `nats_gen` be implemented in the *pure* subset of OCaml (that is, the part without any side effects)? Explain succinctly what aspect of the behavior of `nats_gen` accounts for your answer.

Enter your answer here

Save Answer

Q2.3

5 Points

Define a generator `square_gen : unit -> int` that generates the sequence of perfect squares starting with 1, 4, 9, 16,...:

```
# square_gen () ;;
- : int = 1
# square_gen () ;;
- : int = 4
# square_gen () ;;
- : int = 9
# square_gen () ;;
- : int = 16
```

Enter your answer here

Save Answer

Q2.4

5 Points

Define a function `stream_of_gen` of type `(unit -> 'a) -> 'a stream` that takes a generator and returns a stream of its values. You can assume that the **`NativeLazyStreams`** module from Lab 15 (see here) is open and available to you. Here's an example of the intended behavior of `stream_of_gen` on a fresh instance of `square_gen`:

```
# first 10 (stream_of_gen square_gen) ;;
- : int list = [1; 4; 9; 16; 25; 36; 49; 64; 81; 100]
```

Enter your answer here

Save Answer

Q2.5

5 Points

Above you've defined generators that can generate an unbounded number of elements. Finite generators, which generate a finite number of elements and then stop, need to signal when there are no more elements to generate. We'll use the convention that when a generator is invoked to generate an element but there are no more elements to generate, it raises the `NoMore` exception, defined by

```
exception NoMore ;;
```

With that convention, define a function `gen_of_list` that takes a list as an argument and returns a generator that generates the elements of the list in order. For instance, it should have the following behavior:

```
# let example = gen_of_list ["first"; "second"; "third"] ;;
val example : unit -> string = <fun>
# example () ;;
- : string = "first"
# example () ;;
- : string = "second"
# example () ;;
- : string = "third"
# example () ;;
Exception: NoMore.
```

Enter your answer here

Save Answer

Q3 Formal semantics

30 Points

Q3.1

5 Points

For each of the expressions below, list all of the free variables in the expression, or specify "none" if the expression has no free variables. (The formal definition of the set of free variables is given as Figure 13.3 in the textbook.) You needn't (and shouldn't) give the derivation, just the answer.

1. `let f = fun x -> x + y in f x`

Enter your answer here

2. `(fun x -> x + 1) 3`

Enter your answer here

3. `let f = fun x -> fun y -> if x < y then y else f y x in f 0 1`

Enter your answer here

4. `(fun x -> x) (fun y -> x)`

Enter your answer here

Enter your answer here

5. `(fun x -> (fun y -> x) x)`

Enter your answer here

Save Answer

Q3.2

8 Points

What are the results of the following substitutions. (The formal definition of substitution is given as Figure 13.4 in the textbook.) You needn't (and shouldn't) give the derivation, just the answer.

1. $(\text{fun y -> x})[x \mapsto 42]$

Enter your answer here

2. $((\text{fun y -> x})\ 21)[x \mapsto 42]$

Enter your answer here

3. $(\text{let x} = x + 1 \text{ in } x + 2)[x \mapsto 42]$

Enter your answer here

4. $(\text{let } x = 5 \text{ in } f\ y)[y \mapsto x + 1]$

Enter your answer here

Save Answer

Q3.3

12 Points

Consider the evaluation of the expression `x := !x * 14` in an environment that maps `x` to location l_1 and a store that maps l_1 to `3`. The derivation as per the rules in Figure 19.4 for lexically-scoped environment semantics with mutable storage would look like the following:

$$\{\mathbf{x} \mapsto l_1\}, \{l_1 \mapsto 3\} \vdash \texttt{x := !x * 14}$$

$$\Downarrow$$

$$\{\mathbf{x} \mapsto l_1\}, \{l_1 \mapsto 3\} \vdash \mathbf{x} \Downarrow \boxed{}^{a}, \boxed{\phantom{\{l_1 \mapsto 3\}}}^{b}$$

$$\{\mathbf{x} \mapsto l_1\}, \{l_1 \mapsto 3\} \vdash \texttt{!x * 14}$$

$$\Downarrow$$

$$\{\mathbf{x} \mapsto l_1\}, \{l_1 \mapsto 3\} \vdash \texttt{!x}$$

$$\Downarrow$$

$$\{\mathbf{x} \mapsto l_1\}, \{l_1 \mapsto 3\} \vdash \mathbf{x} \Downarrow \boxed{}^{c}, \boxed{\phantom{\{l_1 \mapsto 3\}}}^{d}$$

$$\Downarrow \boxed{}^{e}, \boxed{}^{f}$$

$$\{\mathbf{x} \mapsto l_1\}, \{l_1 \mapsto 3\} \vdash 14 \Downarrow \boxed{}^{g}, \boxed{\phantom{\{l_1 \mapsto 3\}}}^{h}$$

$$\Downarrow \boxed{}^{i}, \boxed{}^{j}$$

$$\Downarrow \boxed{}^{k}, \boxed{}^{l}$$

We have placed labeled boxes where the "outputs" of each derivation judgement should go – the output value and output store. Your job is to specify what goes in each box to form a well-formed derivation. We've done several of them for you to give you the idea

several of them for you to give you the idea.

(Note: We've made the sizes of the labeled boxes all uniform, so you shouldn't use the sizes as an indication of the size of the corresponding answer.)

e.

Enter your answer here

f.

Enter your answer here

i.

Enter your answer here

j.

Enter your answer here

k.

Enter your answer here

l.

..

Enter your answer here

Save Answer

Q3.4
5 Points

Construct an expression P such that its semantic derivation (in the empty environment and empty store, using the rules of Figure 19.4) would involve a subderivation of exactly the form from the previous problem, that is, the derivation for $\{\}, \{\} \vdash P \Downarrow \cdots$ has within it a subderivation for the value of `x := !x * 14` in an environment that maps `x` to some location l_1 and a store that maps l_1 to `3`.

Enter your answer here

Save Answer

Q4 Complexity
15 Points

Q4.1
3 Points

In problem set 3, you implemented bignums – integers of arbitrary size – including a set of operations on bignums including multiplication. For multiplication, you implemented the "elementary school" algorithm, which is an $O(n^2)$ algorithm. But in the challenge problem, we mentioned that there was an algorithm whose time complexity has a smaller exponent,

Karatsuba's $O(n^{1.58})$ algorithm.

Python's integer data type is a bignum implementation as well. Python's implementation of bignum multiplication uses the Karatsuba algorithm, but perhaps surprisingly, only for numbers larger than 70 digits. Here's the actual pertinent comment from the Python source code:

```
/* For int multiplication, use the O(N**2) school algorithm unless
 * both operands contain more than KARATSUBA_CUTOFF digits....
 */
```

(`KARATSUBA_CUTOFF` is defined as `70` in the code.)

Now $n^2 \gg n^{1.58}$. Why might Python for time performance reasons still use the $O(n^2)$ elementary school algorithm for multiplying integers?

Enter your answer here

Save Answer

Q4.2

12 Points

Here is a simple sorting algorithm for mutable arrays based on swapping adjacent elements that are out of order.

```
(* swap arr i j -- Modifies `arr` by swapping elements
   at indices `i` and `j` *)
let swap array i j =
  let temp = array.(i) in
  array.(i) <- array.(j);
  array.(j) <- temp ;;

(* sort arr -- Modifies `arr` placing its elements in
   sorted order by `<` *)
let rec sort (arr : 'a array) : unit =
```

```
  for pass = 0 to Array.length arr - 1 do
    for index = 0 to Array.length arr - 2 do
      if arr.(index) > arr.(index + 1) then
        swap arr index (index + 1)
    done
  done ;;
```

and here is a demonstration of it working:

```
# let a = [|5; 2; 1; 4; 3|] ;;
val a : int array = [|5; 2; 1; 4; 3|]
# sort a ;;
- : unit = ()
# a ;;
- : int array = [|1; 2; 3; 4; 5|]
```

Which of the following complexity classes does this algorithm fall within, in terms of the number of elements n in the array being sorted? We use c as an arbitrary positive constant. (You can assume that indexing into and finding the length of an array can be done in constant time.)

- [ ] $O(n)$
- [ ] $O(n^2)$
- [ ] $O(n \log c)$
- [ ] $O(n^3)$
- [ ] $O(n^2 + cn)$
- [ ] $O(2^n)$
- [ ] $O(cn)$
- [ ] $O(cn^2)$
- [ ] $O(n^2 - c)$
- [ ] $O(n \log n)$

- [ ] $O(n \log n)$
- [ ] $O(n^2 + c)$
- [ ] $O(\log n)$

Save Answer

Q5 Object-oriented pokemon

10 Points

The following problems concern an object-oriented implementation of the world of Pokemon ("pocket monsters"), the cultural juggernaut found in video games, films, television series, musicals, and even theme parks. All you need to know is that species of Pokemon have funny names and a life force measured in "hit points" (hp). They can also evolve from one species to another. Here is an implementation of a class type and class to get started.

```
class type poke_type =
  object
    (* get_successor -- Returns the Pokemon that this one evolved
       into (as an option), or `None` if it hasn't (yet) evolved. *)
    method get_successor : poke_type option
    (* set_successor evolved_to -- Sets the successor Pokemon that
       this one `evolved_to`. *)
    method set_successor : poke_type option -> unit
    (* hit penalty -- Updates the hit points by reducing them by
       `penalty`. *)
    method hit : int -> unit
    (* is_dead -- Returns `true` if and only if this Pokemon is
       dead. *)
    method is_dead : bool
    (* evolve -- Evolves the Pokemon to a new one of whatever species
       it evolves to. *)
    method evolve : unit
    (* describe -- Returns a text description of the Pokemon and its
       evolutionary successors. *)
    method describe : string
  end

class pokemon (name : string) (init_hp : int) : poke_type =
  object
```

```
    val name : string = name
    val mutable hp : int = init_hp
    val mutable successor : poke_type option = None
    method get_successor = successor
    method set_successor evolved_to =
      successor <- evolved_to
    method hit penalty =
      hp <- max 0 (hp - penalty)
    method is_dead =
      hp <= 0
    method evolve =
      hp <- 0
    method describe =
      Printf.sprintf "%s [%d] --> %s"
                     name hp (match successor with
                              | None -> ""
                              | Some next -> next#describe)
  end
```

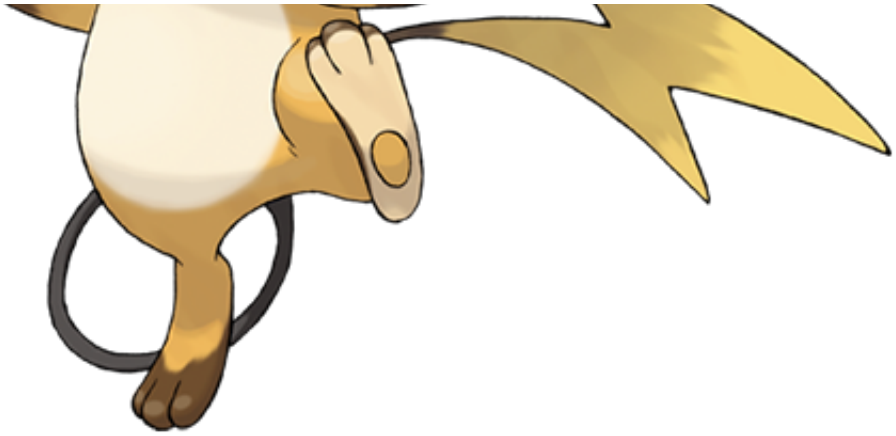
For the purposes here,

- When a Pokemon's hp reaches 0, it dies, as specified in the `is_dead` method.
- When a Pokemon evolves, the Pokemon object's successor is set to the new Pokemon, and the object's hp is set to zero (that is, it dies).

In the next questions, you'll define a few classes for Pokemon species. Because we've given you so much code to get started, *your code should be quite compact, and we'll be looking for succinctness in the code.*

Q5.1

5 Points



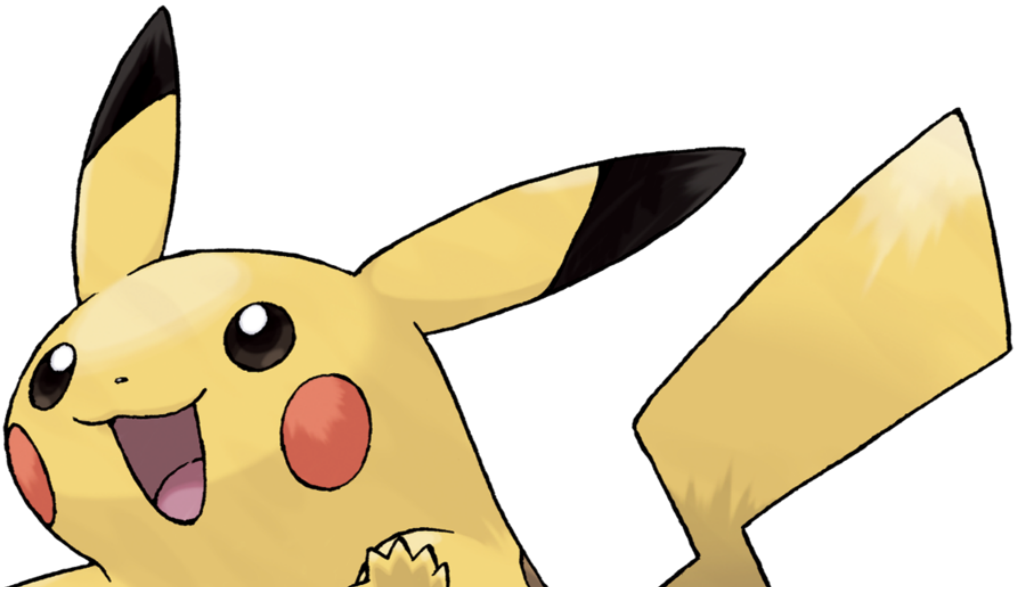
Define a class `raichu` for a Pokemon species whose name is "Raichu" and whose initial hp is 122. Raichu does not evolve into any other kind of Pokemon, so you don't have to worry about that aspect.

Enter your answer here

Save Answer

Q5.2

5 Points



Define a class `pikachu` for a Pokemon species whose name is "Pikachu" and whose initial hp is 82. Pikachu evolves into Raichu, so you'll want to implement that in the `evolve` method.

Enter your answer here

Save Answer

Save All Answers

Submit & View Submission