CSE 341, Autumn 2020, Assignment 4
Due: Thursday, November 12, 5:59PM Pacific time

The starter code contains the module type `INTSET`, copied here.

```
module type INTSET = sig
  type t
  val empty : t
  val contains : int -> t -> bool
  val insert : int -> t -> t
  val remove : int -> t -> t
  val fold : ('a -> int -> 'a) -> 'a -> t -> 'a
  val to_string : t -> string
end
```

This module type defines an abstract data type (ADT) for sets of integers, represented by the type `t` (which is abstract). The provided values and types are mostly intuitive, but it's worth mentioning that (1) the order of iteration in `fold` is "unspecified", meaning its up to the implementation, and clients cannot assume anything about it; and (2) in `to_string`, the elements are printed in sorted order.

The starter code also contains a strange-looking empty module called `Client`, that uses a feature we have not discussed in class to "take a module as an argument". You do not have to understand what this means to do this homework. Just know that, in the body of `Client`, there is a module `I` in scope that implements the module type `INTSET`.

---

1. In the `Client` module, write a function `sum : I.t -> int` that takes a set of integers and returns their sum. Hint: Use `I.fold`. Sample solution is one line. Note that you will not be able to test your code effectively until you do at least one of problems 5 and 6.
2. In a clearly marked comment near your implementation of `sum`, explain why your implementation returns the right answer no matter what order `I.fold` iterates over the elements of the set.
3. In the `Client` module, write a function `to_list : I.t -> int list` that produces a list of the elements in the set. You need to make a subjective design decision about how to handle the fact that `I.fold` does not specify an iteration order. Sample solution is about 3 lines. Note that you will not be able to test your code effectively until you do at least one of problems 5 and 6.
4. In a clearly marked comment near your implementation `to_list`, document and briefly justify your design decision about how the list elements should be ordered.
5. Complete the module implementation `IntSet1` by replacing the calls to `failwith` with real implementations. Do not modify the provided implementations of `insert` or `contains`. In your implementation of `fold`, follow the order of elements in the list. Hint: Note that `insert` can cause duplicates to be in the list. Carefully consider how each function will handle duplicates, and try not to do any work you don't have to do. Sample solution is about 30 lines.
6. Write a module implementation `IntSet2` that also uses an `int list` as its internal representation, but keeps this list sorted and duplicate-free. Be careful that all your functions that return new sets maintain this invariant. Also, take advantage of these invariants whenever possible in your functions to gain efficiency or concision.
7. In a carefully marked comment near the definition of `C1`, describe an *incorrect* way to implement the `Client` module such that `C1` and `C2` would *not* be equivalent. Also in this comment, include a specific test case that would have different behavior under this incorrect implementation.

8. This as a written question. Write your answers in a comment at the bottom of your `hw4.ml` file.

   Consider the following OCaml code (that does not type check)

   ```
   let f g = (g "hello", g 0)
   let p = f (fun x -> x)
   ```

   (a) Ignore the fact that it does not type check, and suppose evaluate `p` anyway. Does anything bad happen? If so, what and why? If not, what value do we get as a result?

   (b) Explain (precisely but relatively briefly) how OCaml's type inference concludes that this code does not type check. Aim for at most a short paragraph.

9. **Challenge**: Write a module `IntSet3` that also implements `INTSET`, but uses a tree or something else interesting for its internal representation.

10. **Challenge**: Can you think of any way to modify OCaml's type system to type check the code from problem 8? What consequences does your idea have for type inference?

Test your functions: Put your testing code in a second file. We will not grade it, but you must turn it in.

Assessment

Your solutions should be correct, in good style, and use only features we have used in class.

Turn-in Instructions

- Upload your `hw4.ml` and `hw4test.ml` to the CSE 341 Autumn 2020 Gradescope Homework 4 Assignment page.