CS 2041: Practice Exam 3

Fall 2018
University of Minnesota

Exam period: 20 minutes
Points available: 40

Problem 1 (10 pts): Write a function called `tree_aslist` which will convert a binary search of strings to a list of strings in order. Examples of its use are below.

Write your code for `tree_aslist` *here.*

```
# type tree =
  | Empty
  | Node of {data: string; left: tree; right: tree}

# let tree_aslist tree = ... ;;
val tree_aslist : tree -> string list = <fun>

# tree_aslist Empty           (* list for empty *)
- : int list = []

# let t1 = ... ;;                 (* 3-node tree *)
# printf "%s\n" (tree_string t1);;
  1: E
0: C
  1: A
# tree_aslist t1;;      (* list for 3-node tree *)
- : string list = ["A"; "C"; "E"]

# let t2 = ...;;                  (* larger tree *)
# printf "%s\n" (tree_string t2);;
  1: H
0: E
    2: D
  1: C
      3: B
    2: A
# tree_aslist t2;;      (* list for larger tree *)
- : int list = ["A"; "B"; "C"; "D"; "E"; "H"]
```

Problem 2 (5 pts): Show the results of parsing given arithmetic expression. Use the symbolic names for from lecture and lab in the resulting data structure. Indent the results to show the structure of the answer.

Write your parsing results here.

```
parse_expr (lex_string "5-4*9/2+7");;
```

Problem 3 (10 pts): Write code that utilizes A4's `Treeset.Make` functor to create a module for sets of unique pairs of `bool` and `string` elements. Define an interface module called `BoolstrEL` with the required bindings. Remember that element comparison functions should check all parts to determine differences. Use a format as indicated below for the element string function. Call the resulting module `BoolstrSet`.

```
# #mod_use "treemap.ml";;
# #mod_use "treeset.ml";;
# #use "setmods.ml";;
module BoolstrEL :
  ...
module BoolstrSet :
  ...
end
# let set =
  BoolstrSet.add BoolstrSet.empty (true,"Crime");;
val set : ...
# BoolstrSet.to_string set;;
- : string = "[(true,Crime)]"
```

Write your code for the interface module and functor call here.

Background: Perseus Tentree is attempting to write a `remove_items` function which operates on OCaml's standard `Sets`. His code is below in a REPL session but does not seem to actually remove items from the set.

```
# module StrSet = Set.Make(String);;
# let set = ...;;
# to_string set;;
- : string = "[B, C, N, R, T, V]"

# let remove_items set items =
    let rec help list =
      match list with
      | [] -> set
      | item::rest ->
         StrSet.remove item set;
         help rest
    in
    help items
  ;;
Warning 10: this expression should have
type unit:   StrSet.remove item set;
             ^^^^^^^^^^^^^^^^^^^^^^
val remove_items :
StrSet.t -> string list -> StrSet.t = <fun>

# remove_items set ["B";"T";"N";"C"];;

# to_string set;;
- : string = "[B, C, N, R, T, V]"
```

Problem 4 (5 pts): Explain the central problem with the code that Perseus has written and why the compiler is issuing a warning about it.

Problem 5 (10 pts): Write a working version of `remove_items` below. You may make the function directly recursive if this proves useful.