CS 3110 PROBLEM SET 1

*Alex Sommer and Maddie Cripps*

*9/10/14*

TODO:

Go over style guide for code

Double check test cases

Problem 1

(a) type: int , value: 42

(b) type: int , value: 42

(c) Not well-typed. List require elements all posses the same type. -3. is a float and not an integer

(d) type: int list , value: [2;4;6;8;10]

(e) type: (string \* string) list , value: [(“zar”,”doz”)]

(f) type: unit , value: ()

(g) type: int option , value: Some 3110

(h) type: int , value: 1764

(i) Not well-typed. f tries to pass f and 10 as arguments to f, but f is of type int -> int and thus only takes one argument.

(j) type: int , value: 12

Problem 2

(a) (1+1);;

(b) [“hello”; “world”];;

(c)

let first (x : float list) : float =

match x with

| [] -> 0.0

| hd :: tl -> hd in first ;;

(d) let f (opt : int option) : int = 1 in f;;

(e) let f (x:int list) (y:int list) : int list = x in f;;

(f) let f (x: int list -> int list) : int list = [1;2] in f;;

(g)

let f (x:int list) : int list = x in

let g (z:int list) : int list -> int list = f in g;;

Note: OCaml is right-associative and functions can only technically take 1 argument so even for (e) we could write that type as int list -> (int list -> int list) . Thus this provided solution but also the solution to (e) are valid answers.

(h) let f ( (x:int) , (y:char list) ) : (int\*char) list = [(5, ‘c’)] in f;;

(i) let x = {hour=10; minute=10; am\_pm="am"};;

(j) let f (x : time) : int = 1 in f;;

Problem 3

(\*

\* Precondition: takes a 2 term tuple of int lists

\* Postcondition: returns a flattened list that is sorted from 0 to the length of the smaller

\*list with the rest of the unused list appended to the end in its original order

\*)

let rec flatten\_with\_smallest\_head ((a:int list), (b:int list)) : int list =

match (a,b) with

| ([],[]) -> []

| (x, []) -> x

| ([],x) -> x

| (x::t1 , y::t2) ->

if (x < y) then x @ sort(t1 , y::t2)

else y @ sort(x::t1 , t2)