HW 02 Environments

CSE4102 Project 02, Spring, 2018

Sarthak Bhatnagar

2/11/18

Section: 001

Instructor: Jeffrey A. Meunier

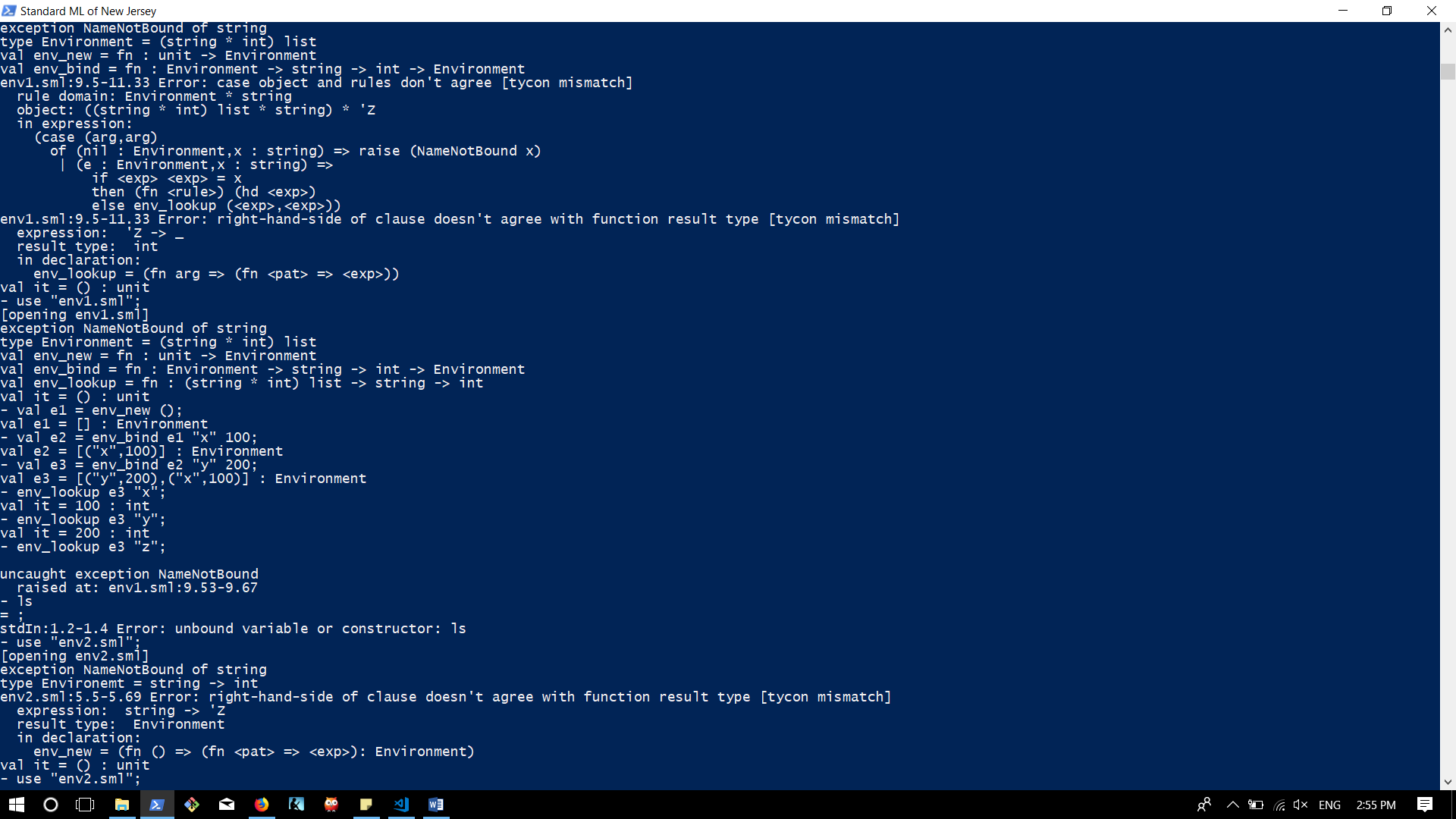
**Introduction**

This homework is a beginning of the us developing an evaluator for a functional programming language over the next few homework assignments.

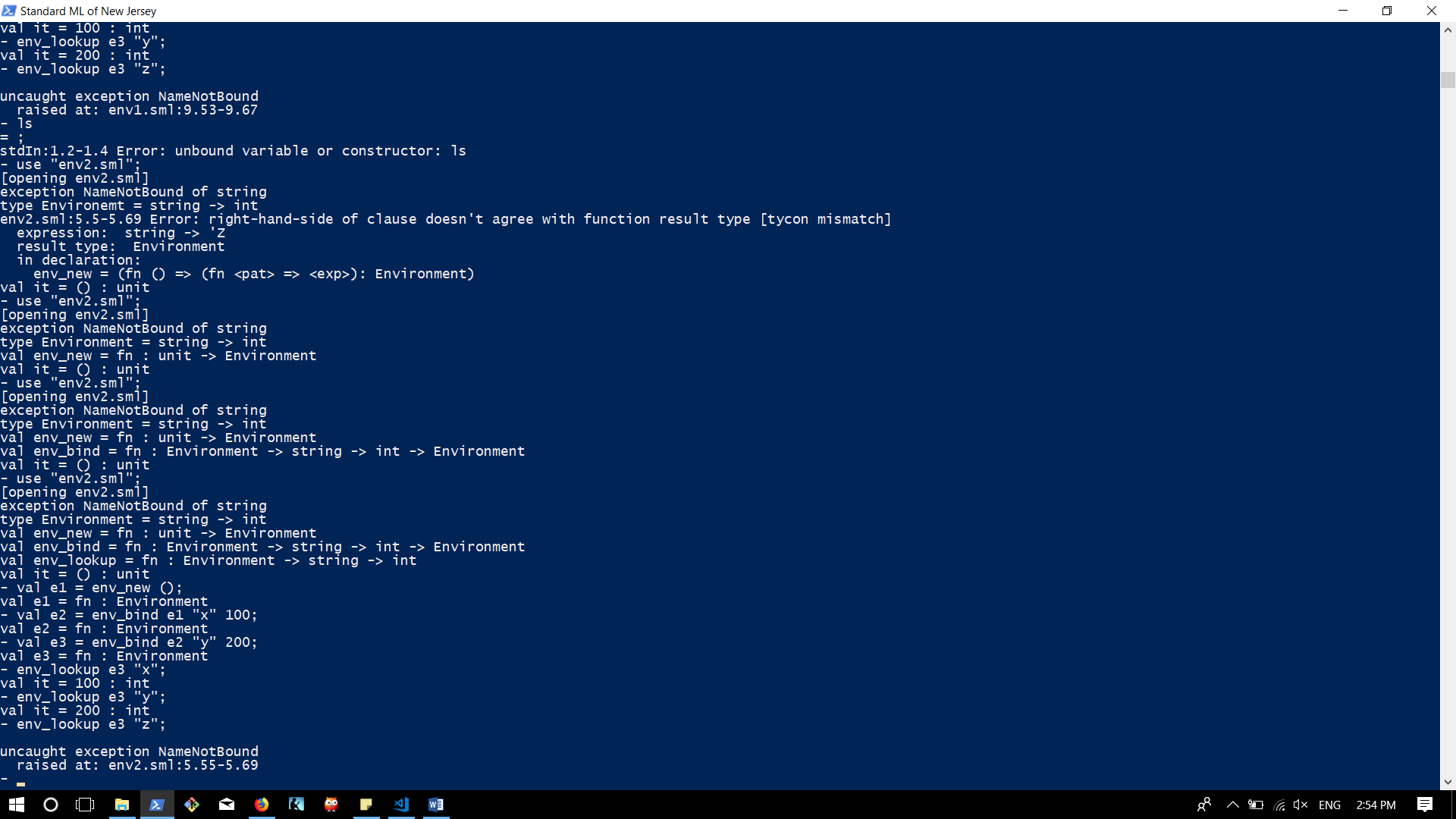
To be able to write interesting programs using this language, we need the ability to bind values to names (i.e., variables). In the evaluator for the language, these names will be managed in a data structure called an environment. The environment is just a list of bindings, where a binding is a mapping of name to value.

We implement the environment in two different ways.

**Output**

Env1.sml

Env2.sml



**Source Code**

env1.sml

(\* crrates an exception if the string is not found \*)

exception NameNotBound of string;

(\* creates an envrionment of type list. the list contains pairs of strings and ints \*)

type Environment = (string \* int)list;

(\* a function that creates a new empty environment \*)

fun env\_new () : Environment = nil;

(\* a function binds a the input values and then adds them to the front of the environemnt \*)

fun env\_bind (e:Environment) (x:string) (y:int) = [(x,y)]@e : Environment;

(\* searches for a string in the list and gives the int value bound to it.

first searches in the hd of the environemnt. if not found, searches the tail \*)

fun env\_lookup (nil:Environment) (x:string) = raise NameNotBound x

|env\_lookup (e:Environment) (x:string) = if (#1(hd e)) = x then #2(hd e)

else env\_lookup(tl e) (x);

env2.sml

(\* creates an exception if the string is not found \*)

exception NameNotBound of string;

(\* creates a type envrionment that maps string to int \*)

type Environment = string -> int;

(\* a function that creates a new environment takes a string argument. if its empty it raises an exception\*)

fun env\_new () : Environment = fn x : string => raise NameNotBound x;

(\* a function that takes in an environment, string, int and then maps that int to string \*)

fun env\_bind (e:Environment) (x:string) (y:int) : Environment = fn (w:string) =>

if w = x then y

else e w;

(\* a function that takes in an environment and a string and return an int bound to the string \*)

fun env\_lookup (e:Environment) (x:string) : int = e x;