Stone Harris
Theory Homework 1
CS320 Concepts of Programming Languages
Due: Sep 27, 11:59 PM

Exercise 1
What is the type of baz (foo bar)?

```
let baz (f : int -> int -> int) (x : int) =
    f x 0

let foo (f : int -> int -> int) (x : int) (y : int) (z : int) =
    f (f x z) (f y z)

let bar (x : int) (y : int) =
    x * y
```

- **baz (foo bar) is of type int -> int -> int**
- **This is because every value being returned by all three of these functions is an int, thus ensuring that the value being returned cuby baz (foo bar) will also be of type of int**

Exercise 2
Define a function that is equivalent to baz (foo bar) but in simpler terms. By equivalent functions, we mean that for all possible inputs, the two functions produce the same output.

let foobarbaz (x : int) (y : int) (z : int) =
    (x * z) * (0 * z)

Exercise 3
Reduce the following expression to a value. Make sure to show all steps.

```
let x = 2 in let y = if x < 0 then true else false in y || false
```

- **let x = 2 in let y = if x < 0 then true else false in y || false**
- **let y = if 2 < 0 then true else false in y || false**
- **false || false**
- **false**

Exercise 4

Reduce the following expression to a value. Make sure to show all steps.

```
let x = 2 in let (l, r) = let x = 3 in (x, x + x) in
if x = l && r = 6 then "abc" else "xyz"
```

- **let x = 2 in let (l, r) = let x = 3 in (x, x + x) in if x = 1 && r = 6 then “abc” else “xyz”**
- **let x = 2 in let (l, r) = let x = 3 in (3, 3 + 3) in if x = 1 && r = 6 then “abc” else “xyz”**
- **let (l, r) = let x = 3 in (3, 6) in if x = 1 && r = 6 then “abc” else “xyz”**
- **let (l, r) = let x = 3 in (3, 6) in false then “abc” else “xyz”**
- **“xyz”**
- **The expression (x = 1 && r = 6) will not be satisfied since x takes on the values of 2 then 3.**
- **Therefore the expression: if x = 1 && r = 6 then “abc” else “xyz” will evaluate to “xyz”**
- **“xyz”**

Exercise 5

Reduce the following expression to a value. Make sure to show all steps.

```
let x = 2 in let y = 1 in
if (let x = 1 in let y = 2 in x = y) then x else y
```

- **let x = 2 in let y = 1 in if (let x = 1 in let y = 2 in x = y) then x else y**
- **let x = 2 in let y = 1 in if (1 = 2) then x else y**
- **if (1 = 2) then 2 else 1**
- **1**
- **When the statement: “if (let x = 1 in let y = 2 in x = y) then x else y” gets evaluated, it will find that x != y so it will not return x, and will return y instead, and y = 1**

Exercise 6

Write a function of the following type:

```
int -> int -> int option
```

that divides two integers. Make sure that this function does not return exceptions for all valid inputs.

let division(x : int) (y : int) : int option -> if y = 0 then None else Some(x/y)

Exercise 7

Sometimes option types are not convenient to use directly as their usage involves a lot of pattern matching. We can write a helper function to help us.

```
let flatMap (opt : int option) (f : int -> int option) =
    match opt with
    | Some n -> f n
    | None -> None
```

What is the type of flatMap?

- **flatMap is of type: int option -> int -> int option -> int option**