Exercise 4: Scope, Type Construction & Memory

# Exercise 4.1: Scope

### Block 1:

Definitions: g

Scope: g function

fun g x =

let

val inc = 1;

fun f y = y + inc;

fun h z =

let

val inc = 2;

in

f z

end

in

h x

end;

### Block 2:

Definitions: inc, f, h

Scope: let (under g) to end;

let

val inc = 1;

fun f y = y + inc;

fun h z =

let

val inc = 2;

in

f z

end

in

h x

end;

### Block 3:

Definitions: inc

Scope: let (under h) to end

let

val inc = 2;

in

f z

end

### Scope of definitions:

* g: block 1
* inc: block 2
* f: block 2
* h: block 2
* inc: block 3

### Occurrences and Binding:

* inc in f y = y + inc: Binds to inc in Block 2.
* f z in Block 3: Binds to f in Block 2.
* h x in Block 2: Binds to h in Block 2.

# Exercise 4.2: Type Construction

datatype suit = Hearts | Diamonds | Clubs | Spades;

fun suitname s = case s of

Hearts => "Hearts"

| Diamonds => "Diamonds"

| Clubs => "Clubs"

| Spades => "Spades";

# Exercise 4.3: Memory

fun f x = x + 1;

1. Yes: The function returns a simple value immediately

fun f x = fn y => x + y;

1. No: The f functions needs x from the second function, so it must stay open

fun f x = fn y => y + 1;

1. Yes: The f function does not need anything from the second function

fun f x = map ~ x;

1. Yes: The f function does not need anything returned from map