**Homework #1**

**Vishal Seenivasan**

**CS 162**

**Winter 2025**

2.1 If you already have an integer, how do you construct a value of type int option?

To use an existing integer (i.e. x = int 5) to construct a value of type int option, we tag the value as Some during assignment: ‘let y = (Some x);;’ would make y of type int option with the value of x.

2.2 If you have nothing at all, how do you construct a value of type int option?

let y = (Some 5);;

2.3 If you have a value of type t (such that you are not told what t is), can you always construct a value of type t option? If so, how? If not, when can you not?

You can always construct a value of type t option, simply specify that the value of type t is tagged as Some and the compiler will recognized the value as type t option.

2.4 If t is some arbitrary type which you have no info about, and you don't have access to any value of type t, can you always construct a value of type t option? If so, how? If not, when can you not?

No – without a value, we cannot specify the type being fed into a’ option – thus the closest type we can achieve would be a’ option.

2.5 If someone gives a t option, can you always consume it to get a value of type t out of it? If so, how? If not, when can you not?

No – when the t option is None, it cannot be consumed to get a type t.

3.1 Given an integer, a boolean, and a string, how do you construct a value of type int \* bool \* string?

let y = (1, true, “string”);;

3.2 Suppose you're given a value of type t1 and a value of type t2 (such that you are not told what t1 and t2 are).

i. Can you always construct a value of type t1 \* t2? If so, how? If not, when can you not?

Yes – for any types t1 and t2, (t1, t2) constructs a list of type t1\*t2.

ii. What about a value of type t2 \* t1?

Yes – same thing, just (t2, t1).

iii. What about t2 \* t1 \* t1 \* t2 \* t1?

Yes (t2, t2, t1, t2, t1).

3.3 If someone gives you a value of type t1 \* t2 (such that you are not told what t1 and t2 are), can you always consume it to get a value of type t1 out of it? Can you always get a value of type t2 out of it? If so, how? If not, when can you not?

Yes to both – Use let (x,y) = p in ...