WRAPPING FUNCTIONS IN A MODULE (10/10 points)

1. Encapsulate the necessary values in a module named `Exp` so that the definition of `example` is accepted by the type checker.

THE GIVEN PRELUDE

```
type e = EInt of int | EMul of e * e | EAdd of e * e
```

YOUR OCAML ENVIRONMENT

```
1  module Exp = struct
2    let int x = EInt x
3
4    let mul a b =
5      match a, b with
6      | EInt 0, _ | _, EInt 0 -> EInt 0
7      | EInt 1, e | e, EInt 1 -> e
8      | a, b -> EMul (a, b)
9
10   let add a b =
11     match a, b with
12     | EInt 0, e | e, EInt 0 -> e
13     | a, b -> EAdd (a, b)
14
15   let rec eval = function
16     | EInt x -> x
17     | EAdd (l, r) -> eval l + eval r
18     | EMul (l, r) -> eval l * eval r
19 end;;
20
21 let example x y z = (* don't change anything to this defintion *)
22   Exp.int (Exp.eval (Exp.mul (Exp.int x) (Exp.add (Exp.int y) (Exp.int z))))
23
```

Evaluate >

Switch >>

Typecheck

Reset Templ

Full-screen |

Check & Sa

Exercise complete (click for details) — 10 pts

v Exercise 1: make `example` compile — Completed, 10 pts

You did not modify `example` , good.
Now, I will check that your function actually works
Found `example` with compatible type.
Computing `example -2 -1 -2`
Correct value `(EInt 6)` — 1 pt
Computing `example 2 -3 -1`
Correct value `(EInt (-8))` — 1 pt
Computing `example 2 4 0`
Correct value `(EInt 8)` — 1 pt
Computing `example 4 -1 1`
Correct value `(EInt 0)` — 1 pt
Computing `example -5 4 0`
Correct value `(EInt (-20))` — 1 pt
Computing `example -1 4 1`
Correct value `(EInt (-5))` — 1 pt
Computing `example 4 0 0`
Correct value `(EInt 0)` — 1 pt
Computing `example -3 3 2`
Correct value `(EInt (-15))` — 1 pt
Computing `example -4 4 -1`
Correct value `(EInt (-12))` — 1 pt
Computing `example 0 -3 4`
Correct value `(EInt 0)` — 1 pt