Assignment 2 (Written) Solutions

Typing in OCaml (1)

There is such a type, we can take τ to be:

```
((int -> int) -> int -> int) -> (int -> int) -> int
```

Explanation: *(not required for full credit)* Since `y z` appears as an operand of integer addition, it must be an `int` in the body of the given function. Since `y` appears as the left-hand-side of an application, it must be a function (whose output type is `int`). Since `y` is applied to `z`, the input type of `y` must be `int` as well. Therefore, `y` is a function of type `int → int`. We can determine the type of `x` in the body of the given function because it is applied to both `y` and `z`, which indicates it is of type `(int → int) → int → int`. We get the above type because `x` and `y` are parameters of anonymous functions.

Typing in OCaml (2)

There is no such type. Since `x` appears as an operand to integer addition, it must be of type `int` in the body `f`. And since `x` is also the argument to the (recursive) function `f`, it must be that `f` is a function whose input type is `int`. Since the body of `f` is defined to be the result of applying `g`, which has type `int → bool`, it must be that the output type of `f` is `bool`, which tells us that `f` has type `int → bool`. But since the arugment to `g` is `f (x + 1)`, the result of applying `f`, it must be that the output type of `f` is `int`, the input type of `g`, indicating that the type of `f` is `int → int`. In short, the output type of `f` is required to be both `int` and `bool`, which is not possible.

For Loops

$$\frac{\Gamma \vdash e_1 : \texttt{int} \qquad \Gamma, \texttt{acc} : \tau \vdash e_2 : \tau \qquad \Gamma \vdash e_3 : \tau}{\Gamma \vdash \texttt{repeat}\ e_1\ \texttt{times}\ e_2\ \texttt{from}\ e_3 : \tau}\ (\mathsf{repeat})$$