2 Foundations of Computer Science (avsm2+jjl25)

The following type definition allows the representation of some mathematical expressions as an OCaml value:

```
type expr =
  | Add of expr * expr
  | Mul of expr * expr
  | Number of int
```

(*a*) Write the OCaml value that corresponds to the expression `(1+4)*(10+2)`. [2 marks]

(*b*) Write a function that will evaluate the numerical result of an `expr` argument. What is the OCaml type of your function? [4 marks]

(*c*) Another way to represent these expressions is to use "Polish notation". In this notation, the operator precedes its operands, and unlike the usual infix notation for expressions there is no need for parentheses. For example, the expression in part (*a*) would be written:

```
*  +  1  4  +  10  2
```

This can be reduced by the following steps:

```
*  5  +  10  2
*  5  12
60
```

Define a type `t` that could be used in a list to represent any expression that can be described with type `expr` above. [2 marks]

(*d*) Write a function `reduce` that performs one step of reduction. Give an example of applying `reduce` to the sample input above twice. *Hint: it should be able to reproduce the steps above.* [8 marks]

(*e*) Write a function `reduce_all` that reduces an argument of type `t list` until it cannot be simplified any further, and returns the simplest value. [4 marks]