Assignment 1

Alessandro Puccia 547462

The permissions that are possible to use inside the language are defined in a new type `perm` that can have values in: `Read`, `Write`, `P1`, `P2`, `P3` and `P4`.

To represent the set of permissions I used the `Set` module offered by the OCaml API instantiated with the new type `perm` introduced (`PSet`). In this way, the basic operations, like the intersection, are not reimplemented. Some examples are provided with different permissions and inherited contexts.

Syntactic constructs modified/added

Given the base interpreter of the previous examples, a new syntactic construct called `DemandPerm` was added. It takes a `SPSet` of permissions S and an expression for which is requested to check if the permissions specified by S are owned by the caller of the function or are inherited.

The `Fun` construct was modified to introduce when defining a function the set of permissions that it requires.

Value constructs modified

The only value construct that was modified is `Closure` to add the set of permissions that the function requires.

Changes to the `eval` interpreter

Another argument is added to the `eval` interpreter to represent the permissions that are inherited and that will be modified by function calls.

The `Fun` case of the pattern matching was only modified to add the set of permissions when creating the closure.

The `Call` case of the pattern matching was changed because, when evaluating the function body, the context permissions will be potentially reduced when computing the intersection between the called function permissions and the actual ones.

Another case is added to implement the `DemandPerm` construct. This construct simply checks that the set of permissions passed (the permissions that are demanded) is a subset of the actual permissions. If this condition holds then the following expression will be evaluated otherwise, an exception is thrown.