Assignment 1

Alessandro Puccia 547462

To represent the permissions I used the Set module offered by the OCaml API istantiated with the string type (`SSet`), 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 `SSet` of permissions S and an expression that requires to check if the permissions specified by S are owned by the caller of the function or are inherited.

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

Value construct modified

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

Changes to the eval interpreter

The interpreter is modified in order to receive another parameter that represents 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 potentially will be reduced computing the intersection between the function permissions and the actual ones.

Another case is added in order to implement the `DemandPerm` construct. This construct simply checks that the set of permissions passed is a subset of the actual permissions. If this condition holds then the following expression will be evaluated otherwise, an exception is launched indicating the permissions that were not owned.