Homework report
Design choices

Bigliazzi Pierfrancesco 550446

The programming language has been extended with a new type called “label” defined as below:

```
type label = Public | Private;;
```

This label has been used to distinguish those variables and functions that can be executed by mobile code or not.
If a variable or a function is Private then it cannot be executed by mobile code otherwise it can.
Then we extend the language by adding security permissions that allow the execution of specific primitives. Below the mapping between permissions and primitives.

Primitive	Permission required
ReadData	Read
WriteData	Write
OpenFile	Open
SendData	Send

Permissions are represented as a new specific algebraic data type called:

```
type perm = Read | Write | Open | Send;;
```

To store the set of permissions of each function at run time I defined a type “permStack” which is a list of lists of permissions:

```
type permStack = (perm list) list;;
```

When an expression that represents mobile code is evaluated with Exec we need to check if there are some variables or functions inside the expression that can be executed or not. To do this I defined a supporting recursive function called “show_label” that checks all the subexpressions inside the expression that has to be evaluated.

```
let rec show_label (e : expr) (env : ‘v t) : label
```

Then inside the Exec primitive in the interpreter if show_label returns a Private label then the mobile is not executed otherwise of all the labels returned are Public the expression can be evaluated.
If the mobile code can be evaluated then I check if all the functions that require Read, Write, Open or Send operations inside the expression have the right permissions by applying a stack inspection algorithm.

The stack inspection algorithm is also applied even if the code executed is local and not mobile. To check the list of permissions inside the stack I used a function called “check_perm”.

```
let rec check_perm (p_stack : permStack) (p : perm) (cond : bool) : bool
```

To check the stack during the evaluation phase the “eval” function has been equipped with “p_stack : permStack” parameter.