Operations on type never

The type `never` is the type of forbidden values. The most prominent scenario in which `never` is used is when implementing a contract template with no additional entrypoint. A contract template defines a set of basic entrypoints, and its `parameter` declaration contains a type variable for additional entrypoints in some branch of an union type, or wrapped inside an option type. Letting this type variable be `never` in a particular implementation indicates that the contract template has not been extended, and turns the branch in the code that processes the additional entrypoints into a forbidden branch.

Values of type `never` cannot occur in a well-typed program. However, they can be abstracted in the `parameter` declaration of a contract—or by using the `LAMBDA` operation—thus indicating that the corresponding branches in the code are forbidden. The type `never` also plays a role when introducing values of union or option type with `LEFT never`, `RIGHT never`, or `NONE never`. In such cases, the created values can be inspected with the operations `IF_LEFT`, `IF_RIGHT`, or `IF_NONE`, and the corresponding branches in the code are forbidden branches.

- `NEVER`: Close a forbidden branch.

```
:: never : 'A  ->  'B
```

- `COMPARE`: Trivial comparison on type `never`

```
:: never : never : 'S   ->   int : 'S
```