UNRAVELING THE AUTOMATIC GRADER (60/60 points)

In this exercise, we will unveil the basics of the grading system.

Note: This exercise is about exceptions, but it uses the `unit` type that is only presented in the next sequence.

1. For each question, we call both your function and a reference function on a list of randomly sampled test cases, and observe the results. We also have to handle the case where a function raises an exception instead of producing a result. Sometimes, we even expect your function to raise an exception, and want to compare it to the exception raised by the reference function.
   For this, we use the `'a result` type given in the prelude.
   Define a function `exec: ('a -> 'b) -> 'a -> 'b result`, that calls the given function on the given argument, and returns `Ok` with the result if everything went well, and `Error` with the exception raised, if there was one.

2. To be able to provide you with the nice error reports, we use an intermediate type for producing reports, similar to the one given in the prelude.
   Define a function `compare` with the following signature.

   ```
   compare : 'a result -> 'a result -> ('a -> string) -> message
   ```

   This function will take first the user function's result and then the reference function's result. It also takes a printer compatible with the return type, to display the results as in one the following cases.

   - `("got correct value 13", Successful)`
   - `("got unexpected value 13", Failed)`
   - `("got correct exception Exit", Successful)`
   - `("got unexpected exception Exit", Failed)`

   You must respect the exact wording for your solution to be accepted. To display exceptions, you can use the provided `exn_to_string` function.

3. Then we define random samplers for each type of data that will be passed to your function. For a given type `'a`, a random sampler simply has type `unit -> 'a`, an imperative function that returns a new value of type `'a` every time you give it a unit.
   Define a function `test` with the following signature.

   ```
   test : ('a -> 'b) -> ('a -> 'b) -> (unit -> 'a) -> ('b -> string) -> report
   ```

   This function must proceed to exactly 10 tests, calling the sampler each time, and return the list of messages. For each sample, you will `exec` the user function (the first parameter), then `exec` the reference function, and `compare` the `result`s. It will then return the list containing the 10 comparison `message`s.
   Your solution must respect the constraint that the first call to the sampler corresponds to the first message of the list, the second to the second, etc. Be cautious about not reversing the list. And since the sampler is an imperative, remember to use `let` ... `in` constructs if necessary, to force the order of evaluation.

THE GIVEN PRELUDE

```
type report = message list
and message = string * status
and status = Successful | Failed

type 'a result = Ok of 'a | Error of exn
```

YOUR OCAML ENVIRONMENT

```
;;

let compare user reference to_string = match user, reference with
  | Ok a, Ok b -> if a = b then ("got correct value " ^ to_string a,Successful)
     else ("got unexpected value " ^ to_string a, Failed)
  | Error a, Error b -> if a=b then ("got correct exception " ^ exn_to_string a,Successful)
     else ("got unexpected exception " ^ exn_to_string a, Failed)
  | Error a, Ok _ -> ("got unexpected exception " ^ exn_to_string a, Failed)
  | Ok a, Error _ -> ("got unexpected value " ^ to_string a, Failed)
;;


let test user reference sample to_string =
  let rec append resultat size =
    if size = 0 then resultat else
      let s = sample () in
      append (resultat @ [compare (exec user s) (exec reference s) to_string]) (size -1)
  in append [] 10
;;
```


Exercise complete (click for details) — 60 pts

v Exercise 1: exec — Completed, 20 pts

Found exec with compatible type.
Computing exec fail_on_odd 3
Correct value (Error Fail) — 1 pt
Computing exec fail 1
Correct value (Error Fail) — 1 pt
Computing exec fail_on_even 1
Correct value (Ok 1) — 1 pt
Computing exec fail_on_odd 3
Correct value (Error Fail) — 1 pt
Computing exec fail_on_even 5
Correct value (Ok 5) — 1 pt
Computing exec fail 4
Correct value (Error Fail) — 1 pt
Computing exec fail 2
Correct value (Error Fail) — 1 pt
Computing exec fail_on_even 3
Correct value (Ok 3) — 1 pt
Computing exec fail_on_odd 3
Correct value (Error Fail) — 1 pt
Computing exec fail 2
Correct value (Error Fail) — 1 pt
Found exec with compatible type.
Computing exec isdigit '5'
Correct value (Ok true) — 1 pt
Computing exec islowercase '1'
Correct value (Error Out_of_range) — 1 pt
Computing exec isuppercase '1'
Correct value (Error Out_of_range) — 1 pt
Computing exec islowercase '1'
Correct value (Error Out_of_range) — 1 pt
Computing exec isuppercase '1'
Correct value (Error Out_of_range) — 1 pt
Computing exec isdigit 'U'
Correct value (Ok false) — 1 pt
Computing exec isdigit 'U'
Correct value (Ok false) — 1 pt
Computing exec isdigit '1'
Correct value (Ok true) — 1 pt
Computing exec isuppercase 'U'
Correct value (Ok true) — 1 pt
Computing exec islowercase '5'

Found compare with compatible type.
Computing compare (Error Out_of_range) (Ok '3') string_of_char
Correct value ("got unexpected exception Out_of_range", Failed) 1 pt
Computing compare (Error Out_of_range) (Error Fail) string_of_char
Correct value ("got unexpected exception Out_of_range", Failed) 1 pt
Computing compare (Error Fail) (Error Out_of_range) string_of_char
Correct value ("got unexpected exception Fail", Failed) 1 pt
Computing compare (Error Out_of_range) (Ok 'Z') string_of_char
Correct value ("got unexpected exception Out_of_range", Failed) 1 pt
Computing compare (Error Out_of_range) (Ok 'a') string_of_char
Correct value ("got unexpected exception Out_of_range", Failed) 1 pt
Computing compare (Error Fail) (Ok 'k') string_of_char
Correct value ("got unexpected exception Fail", Failed) 1 pt
Computing compare (Error Out_of_range) (Error Out_of_range) string_of_char
Correct value ("got correct exception Out_of_range", Successful) 1 pt
Computing compare (Error Out_of_range) (Ok '1') string_of_char
Correct value ("got unexpected exception Out_of_range", Failed) 1 pt
Computing compare (Error Fail) (Error Fail) string_of_char
Correct value ("got correct exception Fail", Successful) 1 pt
Computing compare (Error Out_of_range) (Error Fail) string_of_char
Correct value ("got unexpected exception Out_of_range", Failed) 1 pt
Found compare with compatible type.
Computing compare (Ok 1) (Ok 0) string_of_int
Correct value ("got unexpected value 1", Failed) 1 pt
Computing compare (Error Out_of_range) (Ok 1) string_of_int
Correct value ("got unexpected exception Out_of_range", Failed) 1 pt
Computing compare (Error Out_of_range) (Error Out_of_range) string_of_int
Correct value ("got correct exception Out_of_range", Successful) 1 pt
Computing compare (Ok 2) (Ok 5) string_of_int
Correct value ("got unexpected value 2", Failed) 1 pt
Computing compare (Ok 0) (Error Fail) string_of_int
Correct value ("got unexpected value 0", Failed) 1 pt
Computing compare (Error Out_of_range) (Error Fail) string_of_int
Correct value ("got unexpected exception Out_of_range", Failed) 1 pt
Computing compare (Error Out_of_range) (Error Fail) string_of_int
Correct value ("got unexpected exception Out_of_range", Failed) 1 pt
Computing compare (Ok 5) (Ok 0) string_of_int
Correct value ("got unexpected value 5", Failed) 1 pt
Computing compare (Ok 5) (Error Fail) string_of_int
Correct value ("got unexpected value 5", Failed) 1 pt
Computing compare (Error Out_of_range) (Error Out_of_range) string_of_int
Correct value ("got correct exception Out_of_range", Successful) 1 pt

v Exercise 3: test

Completed, 20 pts

Found test with compatible type.
Computing test islowercase isuppercase sample_char string_of_bool
Correct value 1 pt

```
  [("got correct exception Out_of_range", Successful);
   ("got correct exception Out_of_range", Successful);
   ("got correct exception Out_of_range", Successful);
   ("got unexpected value false", Failed);
   ("got unexpected value true", Failed);
   ("got correct exception Out_of_range", Successful);
   ("got correct exception Out_of_range", Successful);
   ("got correct exception Out_of_range", Successful);
   ("got correct exception Out_of_range", Successful);
   ("got correct exception Out_of_range", Successful)]
```

Computing test islowercase isuppercase sample_char string_of_bool
Correct value 1 pt

```
  [("got correct exception Out_of_range", Successful);
   ("got correct exception Out_of_range", Successful);
   ("got correct exception Out_of_range", Successful);
   ("got unexpected value false", Failed);
   ("got unexpected value true", Failed);
   ("got unexpected value true", Failed);
   ("got unexpected value false", Failed);
   ("got unexpected value true", Failed);
   ("got correct exception Out_of_range", Successful);
   ("got correct exception Out_of_range", Successful)]
```

Computing test islowercase isuppercase sample_char string_of_bool

```
   ("got correct exception Out_of_range", Successful);
   ("got unexpected value true", Failed);
   ("got correct exception Out_of_range", Successful);
   ("got unexpected value true", Failed);
   ("got unexpected value false", Failed);
   ("got unexpected value false", Failed);
   ("got correct exception Out_of_range", Successful)]
```

Computing `test isdigit isdigit sample_char string_of_bool`

Correct value 1 pt

```
  [("got correct value false", Successful);
   ("got correct value true", Successful);
   ("got correct value true", Successful);
   ("got correct value true", Successful);
   ("got correct value false", Successful);
   ("got correct value true", Successful);
   ("got correct value false", Successful);
   ("got correct value false", Successful);
   ("got correct value false", Successful);
   ("got correct value false", Successful)]
```

Computing `test isuppercase islowercase sample_char string_of_bool`

Correct value 1 pt

```
  [("got correct exception Out_of_range", Successful);
   ("got correct exception Out_of_range", Successful);
   ("got unexpected value true", Failed);
   ("got correct exception Out_of_range", Successful);
   ("got unexpected value false", Failed);
   ("got unexpected value false", Failed);
   ("got unexpected value false", Failed);
   ("got unexpected value true", Failed);
   ("got correct exception Out_of_range", Successful);
   ("got correct exception Out_of_range", Successful)]
```

Computing `test isdigit islowercase sample_char string_of_bool`

Correct value 1 pt

```
  [("got unexpected value true", Failed);
   ("got unexpected value false", Failed);
   ("got correct value false", Successful);
   ("got unexpected value true", Failed);
   ("got unexpected value true", Failed);
   ("got correct value false", Successful);
   ("got correct value false", Successful);
   ("got correct value false", Successful);
   ("got unexpected value true", Failed);
   ("got unexpected value false", Failed)]
```

Computing `test isdigit isuppercase sample_char string_of_bool`

Correct value 1 pt

```
  [("got unexpected value false", Failed);
   ("got unexpected value false", Failed);
   ("got correct value false", Successful);
   ("got correct value false", Successful);
   ("got unexpected value true", Failed);
   ("got correct value false", Successful);
   ("got unexpected value true", Failed);
   ("got correct value false", Successful);
   ("got unexpected value false", Failed);
   ("got unexpected value false", Failed)]
```

Computing `test islowercase isdigit sample_char string_of_bool`

Correct value 1 pt

```
  [("got unexpected exception Out_of_range", Failed);
   ("got unexpected value true", Failed);
   ("got unexpected exception Out_of_range", Failed);
   ("got unexpected exception Out_of_range", Failed);
   ("got unexpected exception Out_of_range", Failed);
   ("got unexpected value true", Failed);
   ("got unexpected value true", Failed);
   ("got correct value false", Successful);
   ("got unexpected value true", Failed);
   ("got correct value false", Successful)]
```

Computing `test islowercase isdigit sample_char string_of_bool`

Correct value 1 pt

```
  [("got unexpected value true", Failed);
   ("got correct value false", Successful);
   ("got unexpected value true", Failed);
   ("got correct value false", Successful);
   ("got unexpected exception Out_of_range", Failed);
   ("got unexpected value true", Failed);
   ("got unexpected value true", Failed);
   ("got unexpected exception Out_of_range", Failed);
   ("got correct value false", Successful);
   ("got unexpected exception Out_of_range", Failed)]
```

Computing `test isuppercase isuppercase sample_char string_of_bool`

Correct value 1 pt

```
   ("got correct value true", Successful);
   ("got correct value false", Successful);
   ("got correct value false", Successful);
   ("got correct exception Out_of_range", Successful);
   ("got correct value false", Successful);
   ("got correct value false", Successful)]
```

Found test with compatible type.

Computing test fail_on_even fail_on_odd sample_int string_of_int

Correct value 1 pt

```
  [("got unexpected value 3", Failed); ("got unexpected value 1", Failed);
   ("got unexpected exception Fail", Failed);
   ("got unexpected value 3", Failed);
   ("got unexpected exception Fail", Failed);
   ("got unexpected value 1", Failed);
   ("got unexpected exception Fail", Failed);
   ("got unexpected value 5", Failed); ("got unexpected value 5", Failed);
   ("got unexpected exception Fail", Failed)]
```

Computing test fail_on_even fail_on_odd sample_int string_of_int

Correct value 1 pt

```
  [("got unexpected exception Fail", Failed);
   ("got unexpected exception Fail", Failed);
   ("got unexpected exception Fail", Failed);
   ("got unexpected value 5", Failed); ("got unexpected value 3", Failed);
   ("got unexpected value 3", Failed); ("got unexpected value 3", Failed);
   ("got unexpected exception Fail", Failed);
   ("got unexpected exception Fail", Failed);
   ("got unexpected value 1", Failed)]
```

Computing test fail fail_on_odd sample_int string_of_int

Correct value 1 pt

```
  [("got correct exception Fail", Successful);
   ("got correct exception Fail", Successful);
   ("got unexpected exception Fail", Failed);
   ("got correct exception Fail", Successful);
   ("got correct exception Fail", Successful);
   ("got correct exception Fail", Successful);
   ("got correct exception Fail", Successful);
   ("got correct exception Fail", Successful);
   ("got unexpected exception Fail", Failed);
   ("got correct exception Fail", Successful)]
```

Computing test fail fail_on_even sample_int string_of_int

Correct value 1 pt

```
  [("got unexpected exception Fail", Failed);
   ("got correct exception Fail", Successful);
   ("got correct exception Fail", Successful);
   ("got unexpected exception Fail", Failed);
   ("got correct exception Fail", Successful);
   ("got correct exception Fail", Successful);
   ("got correct exception Fail", Successful);
   ("got unexpected exception Fail", Failed);
   ("got unexpected exception Fail", Failed);
   ("got correct exception Fail", Successful)]
```

Computing test fail_on_odd fail sample_int string_of_int

Correct value 1 pt

```
  [("got correct exception Fail", Successful);
   ("got unexpected value 0", Failed);
   ("got correct exception Fail", Successful);
   ("got unexpected value 2", Failed); ("got unexpected value 4", Failed);
   ("got unexpected value 0", Failed); ("got unexpected value 2", Failed);
   ("got unexpected value 2", Failed); ("got unexpected value 4", Failed);
   ("got correct exception Fail", Successful)]
```

Computing test fail_on_even fail_on_odd sample_int string_of_int

Correct value 1 pt

```
  [("got unexpected exception Fail", Failed);
   ("got unexpected exception Fail", Failed);
   ("got unexpected exception Fail", Failed);
   ("got unexpected value 5", Failed); ("got unexpected value 5", Failed);
   ("got unexpected exception Fail", Failed);
   ("got unexpected value 1", Failed); ("got unexpected value 5", Failed);
   ("got unexpected exception Fail", Failed);
   ("got unexpected exception Fail", Failed)]
```

Computing test fail fail_on_even sample_int string_of_int

Correct value 1 pt

```
  [("got unexpected exception Fail", Failed);
   ("got correct exception Fail", Successful);
   ("got correct exception Fail", Successful);
   ("got correct exception Fail", Successful);
   ("got unexpected exception Fail", Failed);
   ("got correct exception Fail", Successful);
   ("got unexpected exception Fail", Failed);
   ("got unexpected exception Fail", Failed);
   ("got correct exception Fail", Successful);
   ("got unexpected exception Fail", Failed)]
```

Computing test fail_on_odd fail_on_even sample_int string_of_int

```
   ("got unexpected exception Fail", Failed);
   ("got unexpected value 0", Failed);
   ("got unexpected exception Fail", Failed);
   ("got unexpected exception Fail", Failed);
   ("got unexpected value 0", Failed); ("got unexpected value 0", Failed)]
```

Computing `test fail fail_on_even sample_int string_of_int`

Correct value 1 pt

```
  [("got unexpected exception Fail", Failed);
   ("got correct exception Fail", Successful);
   ("got correct exception Fail", Successful);
   ("got unexpected exception Fail", Failed);
   ("got correct exception Fail", Successful);
   ("got correct exception Fail", Successful);
   ("got correct exception Fail", Successful);
   ("got correct exception Fail", Successful);
   ("got correct exception Fail", Successful);
   ("got unexpected exception Fail", Failed)]
```

Computing `test fail fail_on_odd sample_int string_of_int`

Correct value 1 pt

```
  [("got correct exception Fail", Successful);
   ("got correct exception Fail", Successful);
   ("got unexpected exception Fail", Failed);
   ("got unexpected exception Fail", Failed);
   ("got correct exception Fail", Successful);
   ("got correct exception Fail", Successful);
   ("got unexpected exception Fail", Failed);
   ("got unexpected exception Fail", Failed);
   ("got correct exception Fail", Successful);
   ("got unexpected exception Fail", Failed)]
```