CHAR INDEXED HASHTABLES (40/40 points)

Have a look at the documentation of module `Hashtbl`.

1. Implement a module `CharHashedType`, compatible with the `HashedType` signature, where `type t = char`.
2. Use the module defined in the previous question to instantiate the `Hashtbl.Make` functor as a module `CharHashtbl`.
3. Reimplement the data structure of `trie` from a previous exercise, so that a hash table is used to represent the association between characters and children. To do so, complete the definition of module `Trie`, so that it is compatible with the given signature `GenericTrie`, whose `'a table` type is instanciated to `char` indexed hash tables.
   Be careful, a hash table is not a purely functional data structure. Therefore, it must be copied when necessary!
   Note: you must neither change the signature nor the types of module `Trie` or the tests will fail.

THE GIVEN PRELUDE

```
module type GenericTrie = sig
  type 'a char_table
  type 'a trie = Trie of 'a option * 'a trie char_table
  val empty : unit -> 'a trie
  val insert : 'a trie -> string -> 'a -> 'a trie
  val lookup : 'a trie -> string -> 'a option
end
```

YOUR OCAML ENVIRONMENT

```
1  module CharHashedType = struct
2    type t = char
3    let equal k1 k2 = (k1 = k2)
4    let hash k = Hashtbl.hash k
5  end
6
7  module CharHashtbl = Hashtbl.Make (CharHashedType)
8
9  module Trie : GenericTrie
10   with type 'a char_table = 'a CharHashtbl.t =
11 struct
12   type 'a char_table = 'a CharHashtbl.t
13   type 'a trie =  Trie of 'a option * 'a trie char_table
14
15   let empty () = Trie (None, CharHashtbl.create 0)
16
17   let rec lookup trie w =
18     match w, trie with
19     | "", Trie (e, _) -> e
20     | str, Trie (_, t) ->
21         let wbis = (String.sub str 1 (String.length str - 1)) in
22         let result =
23           try
24             CharHashtbl.find t str.[0]
25           with
26             _ -> empty ()
27         in
28         lookup result wbis
29
30   let rec insert trie w k =
31     match w, trie with
32     | "", Trie (e, t) -> Trie (Some k, t)
33     | str, Trie (e, t) ->
```

Evaluate >

Switch >>

Typecheck

Reset Templ

Full-screen

Check & Sa

Exercise complete (click for details) 40 pts

v Exercise 1: `CharHashedType` — Completed, 15 pts

- Found `CharHashedType` with compatible type.
- Bravo! — 5 pts
- Found `CharHashedType.equal` with compatible type.
- Computing `CharHashedType.equal 'x' 'a'`
- Correct value `false` — 1 pt
- Computing `CharHashedType.equal 'g' 'w'`
- Correct value `false` — 1 pt
- Computing `CharHashedType.equal 'w' 'w'`
- Correct value `true` — 1 pt

Correct value `false` 1 pt

Computing `CharHashedType.equal 'n' 'l'`

Correct value `false` 1 pt

Computing `CharHashedType.equal 'b' 'q'`

Correct value `false` 1 pt

Computing `CharHashedType.equal 'p' 'p'`

Correct value `true` 1 pt

Computing `CharHashedType.equal 'p' 'k'`

Correct value `false` 1 pt

Computing `CharHashedType.equal 'i' 'c'`

Correct value `false` 1 pt

v Exercise 2: `CharHashtbl` Completed, 5 pts

Found `CharHashtbl` with compatible type.

Bravo! 5 pts

v Exercise 3: `Trie` Completed, 20 pts

Found `CharHashtbl` with compatible type.

Checking that `Code.Trie.char_table` is compatible with `Code.CharHashtbl.t`

Type found and compatible 5 pts

Found `Trie` with compatible type.

Bravo! 5 pts

Trying to insert
- "roberto" -> -3
- "gregoire" -> 3
- "yann" -> 0
- "cagdas" -> -1
- "ralf" -> 3
- "yann" -> -5
- "ralf" -> -3
- "gregoire" -> 3
- "benjamin" -> -2
- "roberto" -> 4
- "benjamin" -> 1

And then lookup "cagdas"

Correct value (Some (-1)) 1 pt

Trying to insert
- "ralf" -> -4
- "yann" -> 0
- "benjamin" -> -5

And then lookup "gregoire"

Correct value None 1 pt

Trying to insert
- "benjamin" -> 2
- "ralf" -> -3
- "roberto" -> 3

And then lookup "gregoire"

Correct value None 1 pt

Trying to insert
- "benjamin" -> 0
- "yann" -> -4
- "roberto" -> -4
- "cagdas" -> 3
- "gregoire" -> -4
- "ralf" -> 3

And then lookup "cagdas"

Correct value (Some 3) 1 pt

Trying to insert
- "ralf" -> 1
- "cagdas" -> 1
- "roberto" -> 3
- "benjamin" -> -1
- "ralf" -> 2
- "yann" -> 3
- "yann" -> -2
- "cagdas" -> -2
- "gregoire" -> -3
- "ralf" -> -1

And then lookup "roberto"

Correct value (Some 3) 1 pt

Trying to insert

```
And then lookup "roberto"
```
Correct value `None` 1 pt

```
Trying to insert
   - "roberto" -> 2
   - "yann" -> 2
   - "roberto" -> 3
   - "benjamin" -> 3
   - "gregoire" -> 0
   - "yann" -> -2
   - "benjamin" -> 3
   - "gregoire" -> 0
   - "ralf" -> -3

And then lookup "yann"
```
Correct value `(Some (-2))` 1 pt

```
Trying to insert
   - "ralf" -> -4
   - "roberto" -> -1
   - "ralf" -> -2
   - "cagdas" -> 2
   - "benjamin" -> -4
   - "gregoire" -> -3
   - "yann" -> 4
   - "cagdas" -> -5

And then lookup "cagdas"
```
Correct value `(Some (-5))` 1 pt

```
Trying to insert
   - "ralf" -> -1
   - "roberto" -> 0
   - "yann" -> 1
   - "roberto" -> 4
   - "cagdas" -> -4
   - "yann" -> 4
   - "benjamin" -> -4
   - "benjamin" -> -4
   - "ralf" -> -1
   - "gregoire" -> 3
   - "yann" -> 2

And then lookup "gregoire"
```
Correct value `(Some 3)` 1 pt

```
Trying to insert
   - "gregoire" -> 3
   - "yann" -> -3
   - "roberto" -> -4
   - "cagdas" -> 3
   - "cagdas" -> -3

And then lookup "benjamin"
```
Correct value `None` 1 pt