

1995 Paper 13 Question 12

Introduction to Functional Programming

Here is the specification of a dictionary package, based on a type `T`, whose values are data structures that represent finite maps from keys of type `string` to values of type `int`.

```
exception Dict
empty : T
lookup : T * string -> int
update : T * string * int -> T
delete : T * string -> T
merge : T * T -> T
```

Value `empty` represents the empty map. Expression `lookup(t, a)` returns the value bound to the key `a` in `t`, if one exists; otherwise it raises the exception `Dict`. Expression `update(t, a, i)` returns a map that binds key `a` to the value `i`, and on other keys acts the same as `t`. Expression `delete(t, a)` returns the map that is the same as `t` except that key `a` is unbound. Expression `merge(t1, t2)` is a map that binds a key `a` if it is bound in `t1` or `t2`. If it binds `a`, it binds `a` to the value given by `t1`, if it exists, otherwise to the value given by `t2`.

We may represent dictionaries using lists, where `T` is `(string * int) list`. Using this representation, write ML definitions for the values `empty`, `lookup`, `update`, `delete` and `merge`. [8 marks]

Repeat using a functional representation where `T` is `string -> int`. [8 marks]

Discuss the performance of `lookup` in your two implementations. Briefly explain how to obtain a better performance by changing the definition of `T`. [4 marks]