ML exercise
exam examples

Programmazione Funzionale

2024/2025

Exercise

- Write a function `suffixes` (having type `string -> string list`) that given a `string`, returns a list of strings including all the suffixes of the input string (including the string itself and except for the empty string).
- For example:

```
suffixes "ciao" = ["ciao","iao","ao","o"],
suffixes ""= [],
suffixes "hi world" = ["hi world","i world","world","world","orld","rld","ld","d"].
```

Exercise

- Write a function `prefixes` (having type `string -> string list`) that given a `string`, returns a list of strings including all the prefixes of the input string (including the string itself and except for the empty string).
- For example:

```
prefixes "ciao" = ["c" , "ci", "cia", "ciao"];
prefixes "" = [];
prefixes "hi world“ = ["h", "hi", "hi ", "hi
w", "hi wo", "hi wor", "hi worl", "hi world"];
```

Exercise

- Write a function called `sum_binary` (of type `int list * int list -> int list`) that takes as input a pair of lists of integers, which represent a number in binary form, and returns the binary number that represents the sum of the two input numbers, also as an int list of 1s and 0s.
- The use of the "+" operator is **not permitted** within the program, but only logical operators such as "`andalso`", "`orelse`" and the comparison operator "=".
- For example

```
sum_binary([],[]) = []
sum_binary([1,0],[1,0]) =  [1,0,0]
sum_binary([1,0],[0]) = [1,0]
sum_binary([0],[1,0]) =  [1,0]
sum_binary([1,0,0,0],[1,0]) = [1,0,1,0]
sum_binary([1,0,1,1],[1,1,1]) = [1,0,0,1,0]
```

Exercise

- Given the following datatype that defines the structure of a binary tree
  `datatype ctree = Empty | Leaf of char | Node of char * ctree * ctree`
  Write a function `get_words (ctree -> string*string)` that, given a `ctree`, returns a pair of strings, such that the first string is the concatenation of the characters in the leaves and the second string is the concatenation of the characters in the nodes
- For example:

```
get_words(Node (#"a", Leaf #"c", Node (#"b", Leaf #"e", Leaf #"i"))) = (“cei”,”ab”)
```

Exercise

- Given the following datatype that defines the structure of a binary tree

```
datatype Ptree = Empty | PLeaf of int*string |
PNode of int* string * Ptree * Ptree
```

Write a function `count_even (Ptree -> int)` that, given a `Ptree`, returns how many even integers appear as first argument of the pair in the `Ptree`.

- For example:

```
count_even(PNode(2,"good",PNode(3,"good",PLeaf(6
,"bad"),PLeaf(7,"bad")),
Node(5,"bad",PLeaf(1,"good"),Empty))) = 2
```