Principles of Programming, Fall 2009
Practice 7
Patterns in Function Definition and Type System

Woosuk Lee, Suwon Jang, Sungkeun Cho
Programming Research Lab.@SNU

October 26, 2009

1. Using the function `eval` that you defined in Practice 6-ex4, define a procedure called `integral` that takes one polynomial and two numbers (upper bound and lower bound) as input then calculates definite integral value. In short, function call (integral poly l u) approximates mathematical expression $\int_l^u$ poly. Default chunk size is 0.1.

   ```
   integral: poly -> float -> float -> float
   ```

   The type 'poly' is defined as follow.

   ```
   type poly
   = Add of poly * poly
   | Term of coef * expo
   and coef = int
   and expo = int
   ```

2. Trie is a kind of tree data structures efficient for string search. Trie is used to store an associative array where the keys are usually strings. All the descendants of a node have a common prefix of the string associated with that node, and root is associated with the empty string. Values are normally not associated with every node, only with leaves and some inner nodes that correspond to keys of interes. [1]

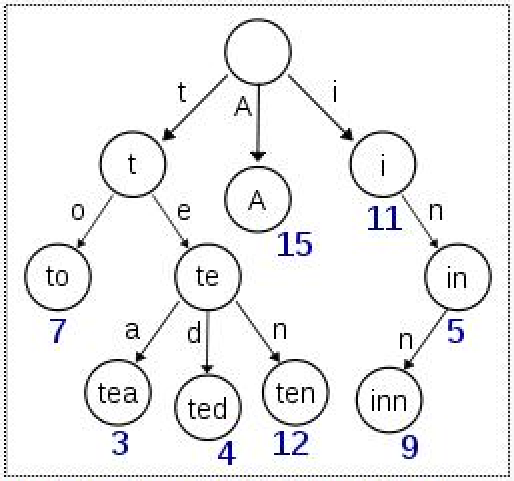
   You can represent a trie type in Ocaml following.

   ```
   type 'a trie = Empty | Node of 'a option * (char * 'a trie) list
   ```

   The type option is given as follow (default).

   ```
   type 'a option = None | Some of 'a
   ```

[1] Excerted from Wikipedia.



A trie for keys "A", "to", "tea", "ted", "ten", "i", "in", and "inn".

Figure 1: an example of trie

Define following two procedures.

```
insert: 'a trie -> string -> 'a -> 'a trie
lookup: 'a trie -> string -> 'a
```

If you fail to lookup raise the exception `Not_found`(predefined exception).
We recommend you to use `String.get` and `String.sub`.