Homework 2
ENE4014 Programming Languages, Spring 2020
Woosuk Lee
due: 4/28(Tue), 24:00

Exercise 1 Write a function

$$\texttt{calculate} : \texttt{exp} \rightarrow \texttt{float}$$

that returns a result of a given arithmetic formula. The `exp` type is defined as follows:

```
type exp = X | INT of int
  | REAL of float
  | ADD of exp * exp
  | SUB of exp * exp
  | MUL of exp * exp
  | DIV of exp * exp
  | SIGMA of exp * exp * exp
  | INTEGRAL of exp * exp * exp
```

For example, the following arithmetic formulas can be written in the exp type:

$\sum_{x=1}^{10}(x \times x - 1)$	`SIGMA(INT1, INT10, SUB(MUL(X, X), INT1))`
$\int_{x=1.0}^{10.0}(x \times x - 1)dx$	`INTEGRAL(REAL1.0, REAL10.0, SUB(MUL(X, X), INT1))`

When you compute integrals, dx should be 0.1. □

Exercise 2 Write a function

$$\texttt{diff} : \texttt{ae} * \texttt{string} \rightarrow \texttt{AE}$$

that differentiates the given algebraic expression with respect to the variable given as the second argument. The `ae` type is defined as follows:

```
type ae = CONST of int
  | VAR of string
  | POWER of string * int
  | TIMES of ae list
  | SUM of ae list
```

For example, $x^2 + 2x + 1$ is represented by

```
SUM [POWER ("x", 2); TIMES [CONST 2; VAR "x"]; CONST 1]
```

and differentiating it (w.r.t. "x") gives $2x + 2$, which can be represented by

```
SUM [TIMES [CONST 2; VAR "x"]; CONST 2]
```

□

Exercise 3 Write a function

$$\texttt{to\_list} : \texttt{string} \to \texttt{char list}$$

which transforms a given string into a list of characters. For instance,

$$\texttt{to\_list}\ \text{“}\texttt{ocaml}\text{”} = [\texttt{'o'};\texttt{'c'};\texttt{'a'};\texttt{'m'};\texttt{'l'}]$$
$$\texttt{to\_list}\ \text{“}\texttt{PL}\text{”} = [\texttt{'P'};\texttt{'L'}]$$
$$\texttt{to\_list}\ \text{“”} = []$$

You may refer to `https://caml.inria.fr/pub/docs/manual-ocaml/libref/String.html` to find out any useful string operations. For example, `String.get ''abc'' 0` returns `'a'`. □

Exercise 4 Write a function

$$\texttt{repeat} : \texttt{string} \to \texttt{int} \to \texttt{string}.$$

`repeat s n` returns a string obtained by concatenating `s` `n` times. For instance,

$$\texttt{repeat}\ \text{“}\texttt{ocaml}\text{”}\ 3 = \text{”}\texttt{ocamlocamlocaml}\text{”}$$
$$\texttt{repeat}\ \text{”}\texttt{PL}\text{”}\ 2 = \text{”}\texttt{PLPL}\text{”}$$
$$\texttt{repeat}\ \text{“}\texttt{PL}\text{”}\ 0 = \text{””}$$

You may use the (^) operator to concatenate two strings. For example,

```
"a" ^ "b" = "ab"
```

□

Exercise 5 Write a function

$$\texttt{count\_string} : \texttt{string} \to \texttt{string} \to \texttt{int}.$$

that returns the number of substrings in a given string. `count_string s x` count how many times `x` is found in `s` (if `x` is the empty string `""`, `0` is returned. For instance,

$$\texttt{count\_string}\ \text{“}\texttt{aaa}''\ \text{“}\texttt{a}\text{”} = 3$$
$$\texttt{count\_string}\ \text{“}\texttt{aaa}''\ \text{“}\texttt{aa}\text{”} = 2$$
$$\texttt{count\_string}\ \text{“}\texttt{abababababa}''\ \text{“}\texttt{aba}\text{”} = 5$$
$$\texttt{count\_string}\ \text{“}\texttt{GeeksforGeeksforGeeksforGeeks}''\ \text{“}\texttt{GeeksforGeeks}\text{”} = 3$$
$$\texttt{count\_string}\ \text{“}\texttt{aaa}''\ \text{“”} = 0$$

Note that the function should give correct results when two occurrences of the substring overlap. □

Exercise 6 Binary trees can be defined as follows:

```
type btree = Empty | Node of int * btree * btree
```

For example, the following `t1` and `t2` are binary trees.

```
let t1 = Node (1, Empty, Empty)
let t2 = Node (1, Node (2, Empty, Empty), Node (3, Empty, Empty))
let t3 = Node (1, Node (2, Node (3, Empty, Empty), Empty), Empty)
let t4 = Node (1, Node (2, Empty, Empty), Empty)
```

Write a function

$$\texttt{check} : \texttt{btree} \to \texttt{bool}$$

that checks if a given binary tree is balanced. A tree is balanced when for every node the height of the subnodes differs by at most 1. For example,

$$\begin{aligned} \texttt{check Empty} &= \texttt{true} \\ \texttt{check t1} &= \texttt{true} \\ \texttt{check t2} &= \texttt{true} \\ \texttt{check t3} &= \texttt{false} \\ \texttt{check t4} &= \texttt{true} \end{aligned}$$

□

Exercise 7 Write a function

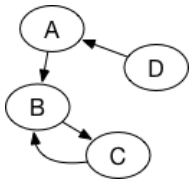
$$\texttt{uniq} : \texttt{'a list} \to \texttt{'a list}$$

which removes duplicated elements from a given list so that the list contains unique elements. For instance,

$$\texttt{uniq}\ [5;6;5;4] = [5;6;4]$$
$$\texttt{uniq}\ [\texttt{'a}';\texttt{'b}';\texttt{'c}';\texttt{'c}'] = [\texttt{'a}';\texttt{'b}';\texttt{'c}']$$

□

Exercise 8 A graph is a structure amounting to a set of nodes in which some pairs of the nodes are related[1]. We can depict personal relationships by a graph. Suppose person A likes person B, who likes person C. Person C also likes person B. Another person D likes person A. These relations can be represented by the following graph.



Let us suppose this "like" relation is *transitive* in the sense that

if A likes B and B likes C, then A also likes C.

Write a function

$$\texttt{likes} : \texttt{relationships} \to \texttt{person} \to \texttt{int}$$

that given a graph of personal relationships and a person, returns the number of people whom the person likes. The type `relationships` is for representing a graph of personal relationships, which is defined as follow:

```
type relationships = string * string list
```

For example, the above graph can be represented as

```
let graph = [("A", "B"); ("B", "C"); ("C", "B"); ("D", "A")]
```

The function should behave as follows:

```
likes graph "A" = 2
likes graph "B" = 2
likes graph "C" = 2
likes graph "D" = 3
```

□

Exercise 9 In the above graph of relationships, the B person likes herself for the following reason: B likes C and C likes B, and because the relation is transitive, B also likes B. For a similar reason, C also likes herself.

Write a function

$$\texttt{selflove} : \texttt{relationships} \to \texttt{int}$$

that returns the number of people who are in self-love. For example, given the above `graph`, `selflove graph` should return 2 because B and C like themselves.
□

[1] https://en.wikipedia.org/wiki/Graph_(discrete_mathematics)