Homework 2
ENE4014 Programming Languages, Spring 2021

Woosuk Lee

due: 4/26(Mon), 24:00

- You must write your code by yourself and must not look at someone elses code. An automatic code clone detector (SW tool) will be used to detect any violation of the rule.
- Do not use any external libraries. You can use only the OCaml standard library (`https://ocaml.org/api/index.html`).

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:

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

When you compute integrals, dx should be 0.1. $\square$

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 The following type specifies a set of *big integers* which can represent positive numbers larger than the 32-bit integer limit ($2^{32} - 1$).

```
type bigint = BigInt of string
```

For example, a big integer 1032 is represented as `BigInt` "1032".

Write a function

```
compute_bigint :  op -> bigint * bigint -> bigint
```

that returns a resulting big integer for a given arithmetic operation along with two big integers as operands, and the `op` type is defined as follows:

```
type op = ADD | MUL
```

The function should behaves as follows:

```
compute_bigint ADD ((BigInt ‘‘1032"), (BigInt ‘‘1032"))
  = BigInt ‘‘2064"
compute_bigint MUL ((BigInt ‘‘183829184395481829391"), (BigInt ‘‘76187828"))
  = BigInt ‘‘14005546282103253594766852748"
```

Exercise 4 Write a function

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

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,

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

Note that the function should give correct results when two occurrences of the substring overlap. You may refer to the OCaml standard API `https://ocaml.org/api/String.html` for useful functions for manipulating strings.

Exercise 5 Consider the following triangle which is called Pascals triangle:

```
        1
      1   1
    1   2   1
  1   3   3   1
1   4   6   4   1
```

where the numbers on an edge of the triangle are all 1, and each number inside the triangle is the sum of the two numbers above it.

Write a function

```
pascal :  int * int -> int
```

that computes an element of Pascals triangle at a location specified by a row and a column (starting from 0). For example, `pascal` should behave as follows:

```
pascal (0,0) = 1
pascal (1,0) = 1
pascal (0,0) = 1
pascal (2,1) = 2
pascal (4,2) = 6
```

Exercise 6 The edit distance[1] is a string metric for measuring the difference between two strings. Informally, the edit distance between two strings is the minimum number of single-character edits (insertions, deletions or substitutions) required to change one string into the other. For example, the edit distance between “kitten” and “sitting” is 3, since the following three edits change one into the other, and there is no way to do it with fewer than three edits:

[1]https://en.wikipedia.org/wiki/Levenshtein_distance

1. kitten $\rightarrow$ sitten (substitution of “s” for “k”)
2. sitten $\rightarrow$ sittin (substitution of “i” for “e”)
3. sittin $\rightarrow$ sitting (insertion of “g” at the end).

Write a function

```
closest : string -> string list -> string
```

which takes a string s and a list of string *vocab*, and returns a string in *vocab* which is the most similar to s. For example,

```
closest "kitten" ["sitting"; "fighting"; "kicking"] = "sitting"
```

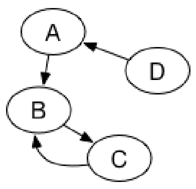
because the edit distance between “kitten” and “”fighting” is 5 and the distance between “kitten” and “kicking” is 4. In case of tie (i.e., multiple strings of the same edit distance), the leftmost element in the list *vocab* should be returned.

When computing the edit distance between two strings a and b (denoted $dist(a, b)$, you may refer to the following inductive definition. $tail(a)$ denotes a substring of a without the first character of a (e.g., $tail(\text{"}abc\text{"}) = \text{"}bc\text{"}$).

$$dist(a,b) = \begin{cases} \text{length of } a & (b \text{ is the empty string}) \\ \text{length of } b & (a \text{ is the empty string}) \\ dist(tail(a), tail(b)) & (\text{the first characters of } a \text{ and } b \text{ are equal}) \\ 1 + \min \begin{cases} dist(tail(a), b) \\ dist(a, tail(b)) \\ dist(tail(a), tail(b)) \end{cases} & \end{cases}$$

Exercise 7 A graph is a structure amounting to a set of nodes in which some pairs of the nodes are related[2].

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

```
likes : relationships -> person -> int
```

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

that given a graph of 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 8 In the above graph, B likes him/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 him/herself.

Write a function

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

that returns the number of people who like him/herself. For example, given the above `graph`, `selflove graph` should return 2.