Homework 3
COSE212, Fall 2016

Hakjoo Oh

Due: 11/4, 24:00

Problem 1 Write a function

```
diff : aexp * string -> aexp
```

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

```
type aexp =
 | Const of int
 | Var of string
 | Power of string * int
 | Times of aexp list
 | Sum of aexp 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]
```

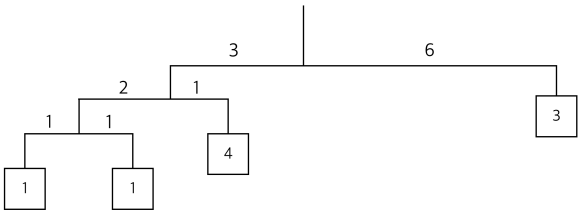
Note that the representation of $2x + 2$ in `aexp` is not unique. For instance, the following also represents $2x + 2$:

```
Sum
 [Times [Const 2; Power ("x", 1)];
  Sum
   [Times [Const 0; Var "x"];
    Times [Const 2; Sum [Times [Const 1]; Times [Var "x"; Const 0]]]];
  Const 0]
```

Problem 2 A binary mobile consists of two branches, a left branch and a right branch. Each branch is a rod of a certain length, from which hangs either a weight or another binary mobile. In OCaml datatype, a binary mobile can be defined as follows:

```
type mobile = branch * branch     (* left and rigth branches *)
and branch = SimpleBranch of length * weight
           | CompoundBranch of length * mobile
and length = int
and weight = int
```

A branch is either a simple branch, which is constructed from a length together with a weight, or a compound branch, which is constructed from a length together with another mobile. For instance, the mobile



is represented by the following:

```
(CompoundBranch (3,
  (CompoundBranch (2, (SimpleBranch (1, 1), SimpleBranch (1, 1))),
   SimpleBranch (1, 4))),
 SimpleBranch (6, 3))
```

Define the function

```
balanced : mobile -> bool
```

that tests whether a binary mobile is balanced. A mobile is said to be *balanced* if the torque applied by its top-left branch is equal to that applied by its top-right branch (that is, if the length of the left rod multiplied by the weight hanging from that rod is equal to the corresponding product for the right side) and if each of the submobiles hanging off its branches is balanced. For example, the example mobile above is balanced.

Problem 3 Consider the following expressions:

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

Implement a calculator for the expressions:

```
calculator : exp -> int
```

For instance,

$$\sum_{x=1}^{10}(x * x - 1)$$

is represented by

```
SIGMA(INT 1, INT 10, SUB(MUL(X, X), INT 1))
```

and evaluating it should give 375.

Problem 4 Consider the following language:

```
type exp = V of var
         | P of var * exp
         | C of exp * exp
and var = string
```

In this language, a program is simply a variable, a procedure, or a procedure call.

Write a checker function

```
check : exp -> bool
```

that checks if a given program is well-formed. A program is said to be *well-formed* if and only if the program does not contain free variables; i.e., every variable name is bound by some procedure that encompasses the variable. For example, well-formed programs are:

- `P ("a", V "a")`
- `P ("a", P ("a", V "a"))`
- `P ("a", P ("b", C (V "a", V "b")))`
- `P ("a", C (V "a", P ("b", V "a")))`

Ill-formed ones are:

- `P ("a", V "b")`
- `P ("a", C (V "a", P ("b", V "c")))`
- `P ("a", P ("b", C (V "a", V "c")))`