CSL100: Introduction to Computers and Computer Science
I semester 2013-14
Instructors: Subhashis Banerjee and S. Arun-Kumar

Assignment 6
SYMBOLIC COMPUTATION

The main purpose of this package is to do symbolic computations, such as differentiation (including partial differentiation) and symbolic integration with names denoting variables or constants. Your task is to implement the following signature (given in SML syntax) in SML or python.

```
signature Algebra =

sig
    exception divide_by_zero
    exception negative_exponent
    exception zero_raised_to_zero
    exception not_variable
    exception not_symbolic_constant
    exception equality_undefined
    exception intvalue_undefined

    datatype Symex = ICOEFF of int
                   | COEFF of string
                   | VAR of string
                   | POWER of Symex * int
                   | NEG of Symex
                   | PLUS of Symex * Symex
                   | MULT of Symex * Symex
    val showSymex  : Symex -> unit
    val symequal   : Symex * Symex -> bool
    val deriv      : Symex -> Symex -> Symex
    val integral   : Symex -> Symex -> Symex
    val valsymex   : Symex * (Symex * int) list -> Symex
    val value      : Symex * (Symex * int) list -> Symex
    val intvalue   : Symex -> Symex
end;
```

Explanations and Notes This datatype is best understood with an example. Consider the cubic $a * x^3 + (4.0 * x^2 + (b * x + c))$ which would be represented as

```
      PLUS (MULT (COEFF ("a"),
                  POWER (VAR ("x"),3)),
            PLUS (MULT  (ICOEFF (4),
                         POWER (VAR ("x"),2)),
                  PLUS  (MULT (COEFF ("b"),
                               VAR ("x")),
                         COEFF ("c"))
```

```
        )
    )
```

- **ICOEFF** represents a integer whereas **COEFF** is a symbolic constant which may later be assigned an integer. In the expression $a * x^3 + (4 * x^2 + (b * x + c))$, it is usually understood that "4" is an integer coefficient whereas "a", "b" and "c" are symbolic coefficients, and "x" is a variable. This quadratic is a polynomial (of degree 2) in the variable "x" with coefficients "a", "4", "b" and "c" respectively.
- The datatype **Symex** denotes the construction of symbolic expressions inductively and allows the use of only integer coefficients as well as symbolic constants.
- The constructor **VAR** defines names that are to be considered variables, so that differentiation and integration can be performed with respect to them.
- The constructor **COEFF** defines names that are to be considered to be "symbolic" constant values. In the example above "a", "b" and "c" are such symbolic constants.
- The constructor **POWER** allows only non-negative integer exponents (otherwise the exception **negative_exponent** is raised) on symbolic expressions.
- **showSymex** displays a symbolic expression in *infix* form using the operators `^, -, +, *, /, (, )`. *Of course, you will have to use parentheses.*
- **symequal** can compare only purely symbolic expressions. For example, it should be able to show that $(a * x + b) * (a * x - b) = a * a * x^2 + b * b$ and $(a * x + b) * (c * x - d) = a * c * x^2 + (c * b - a * d) * x + (b - d)$. But in general equality checking can very tricky unless normal forms are uniquely defined. I hope the following detailed definition of a normal form for the polynomials will be suitable.
- **Normal forms** In case there are integer constants in the expressions then integer expressions are simplified to yield a single integer. For example, differentiating the expression

  $$a * x^3 + (4 * x^2 + (b * x + c))$$

  would yield the expression

  $$3 * a * x^2 + (2 * 4 * x^1 + b)$$

  which on simplification would yield

  $$3 * a * x^2 + (8 * x + b)$$

  This would be the **normal form** of the expression. Hence any expression which is a polynomial is ultimately expressed in terms of a normal form as follows:

  1. If the coefficient of a term reduces to the integer value 0, that term is automatically removed.
  2. If any variable has an exponent of 0, that variable is automatically removed from the term. Similarly if any variable has an exponent of 1, then the exponent is automatically removed and it is no longer a POWER expression; instead it becomes a simple variable.
  3. The operations "PLUS" and "MULT" are *left associative* and "POWER" is *right associative*. Thus PLUS (PLUS (3, 4), 5) **is** in normal form, whereas PLUS (3, PLUS (4, 5)) and PLUS (PLUS (3, 4), PLUS (5, 6)) are **not** in normal form.
  4. *The lexicographic ordering induced by* $<$. Let $<$ be any ordering on primitive elements (such as characters or integers). The lexicographic ordering induced by $<$ on sequences (i.e. lists, tuples, strings etc.) of primitive elements is as follows: For any two sequences `s` and `t`, `s` $<$ `t` if
     - `s` is a *proper prefix* of `t`, or
     - they have identical primitive elements starting from left to right till at some point i, $s_i < t_i$, where s_i is the $i-th$ element of `s` and t_i is the $i-th$ element of `t`.

5. *The "ASCII-betical" ordering.* For any two characters c_1 and c_2, $c_1 < c_2$ if $ord(c_1) < ord(c_2)$. Hence $\mathsf{A} < \ldots < \mathsf{Z} < \mathsf{a} < \ldots < \mathsf{z}$
6. The variable symbols, being strings, are all ordered by the *lexicographic ordering induced by the "ASCII-betical" ordering.* Similarly the symbolic constants also being strings are ordered lexicographically.
7. A *monomial in normal form* is a product of at most one integer followed by the symbolic constants in lexicographic order, followed by the powers of variables again in the lexicographic ordering of the variables.
8. The polynomial expression is expressed as a sum of monomials in normal form. The order of appearance of the monomials is as follows. Let $\{x_1, \ldots, x_n\}$ be the set of all variables in the polynomial such that $x_1 < \ldots < x_n$ in the lexicographic ordering.
   (a) For each monomial, form the n-tuple $\langle p_1, p_2, \ldots p_n \rangle$, where each p_i is the (non-negative) integer exponent of the variable x_i, $1 \leq i \leq n$ in the monomial ($p_i = 0$ if x_i does not occur in the monomial). This is the *degree-tuple* of the monomial. Again use the lexicograpahic ordering on degree-tuples induced by the "<" ordering on integers.
   (b) The monomial with the greatest degree-tuple (in the lexicographic ordering of tuples) is the first term in the sum, and the one with the least degree (usually a constant term) appears last.

- **deriv** defines the (partial) derivative of a symbolic expression with respect to a variable. Note that this is a general purpose function which should raise the exception **not_variable** if its *second* argument is anything other than a variable.
- **integral** similarly defines an *indefinite* integral. *This means that in each session of the use of this package a brand new constant should be generated for each application of this function.* These constants should also be different from whatever constants have been currently defined in the session. The exception **not_variable** is raised if its *second* argument is anything other than a variable.
- **valsymex** computes the symbolic value of a symbolic expression given a (partial) list of ordered pairs of values for various symbolic constants. This value expression should be in *normal form.*
- **value** similarly computes the symbolic value of a possibly multivariate expression given the values of individual variables.
- **intvalue** of a symbolic expression is defined only if the expression is a simple one constructed purely from integer values (and containing no symbolic variables or symbolic constants). Otherwise the exception **intvalue_undefined** is raised.