/* Mathematical operation library

Author - pizmon muirsheendurkin@gmail.com

Decimal long-string arithmetic procedures. Decimal point is allowed,
signs (+/-) are not allowed.

Useful thing for study and reminiscences, not for production releases.

You may use this code without any restrictions, but you must mention me.
MindTheBird - and mind pizmon too.

Процедуры десятичной длинностроковой точной арифметики разработаны
в исторически-юмористически-дидактических целях. Поддерживается
десятичная точка, не поддерживаются знаки (+/-).

Автор подозревает о существовании ряда полезных нововведений,
позволяющих "этим вот" больше не заниматься. Но автор обращает
внимание благосклонной аудитории на следующее:

1. Как не хватало нам этих полезных нововведений в те далекие годы, когда
   Firebird был Interbase, а ОС Novell Netware была тем, чем она была! И
   ничего, почти все выжили.

2. Умение, как говорил шахматист Остап Бендер, "с ничтожными силами овладеть
   всей доской" стимулирует интеллектуальную смелость, необходимую
   для профессии программиста. Изучать древний опыт также полезно.
   "Чтобы стоять, я должен держаться корней".

```
 Допускается любое использование этих процедур при упоминании автора.

 (c) pizmon

set term ^;

 this extracts from string "s" a digit defined by template "templ".
 Nowadays we've got "substring" to do it without kinking our brain.

 эта процедура вычленяет из строки s разряд, определяемый шаблоном templ.
 для этого в наше время используется substring.
*/
CREATE PROCEDURE dig (
    s VARCHAR(1000),
    templ VARCHAR(1000)
)
RETURNS (
    result INTEGER
)
AS
BEGIN
    IF (
        s LIKE
        '%' ||
        '0' ||
        templ) THEN
        result = 0;
    ELSE
        IF (
            s LIKE
            '%' ||
            '1' ||
            templ) THEN
            result = 1;
        ELSE
            IF (
                s LIKE
                '%' ||
                '2' ||
                templ) THEN
```

```
    result = 2;
ELSE
    IF (
        s LIKE
        '%' ||
        '3' ||
        templ) THEN
        result = 3;
    ELSE
        IF (
            s LIKE
            '%' ||
            '4' ||
            templ) THEN
            result = 4;
        ELSE
            IF (
                s LIKE
                '%' ||
                '5' ||
                templ) THEN
                result = 5;
            ELSE
                IF (
                    s LIKE
                    '%' ||
                    '6' ||
                    templ) THEN
                    result = 6;
                ELSE
                    IF (
                        s LIKE
                        '%' ||
                        '7' ||
                        templ) THEN
                        result = 7;
                    ELSE
                        IF (
                            s LIKE
                            '%' ||
                            '8' ||
                            templ) THEN
                            result = 8;
                        ELSE
                            IF (
                                s LIKE
                                '%' ||
                                '9' ||
                                templ) THEN
                                result = 9;
                            ELSE
                                IF (
                                    s LIKE
                                    '%' ||
                                    '.' ||
                                    templ) THEN
```

```
                                                    result = -1;
                                                ELSE
                                                    result = NULL;
    SUSPEND;
END
^



/*

 this divides string "s" by ten shifting decimal point.


 эта процедура делит строку s на 10 методом сдвига десятичной точки.

*/
CREATE PROCEDURE strshift (
    s VARCHAR(1000)
)
RETURNS (
    result VARCHAR(1000)
)
AS
    DECLARE VARIABLE templ VARCHAR(1000);
    DECLARE VARIABLE d INTEGER;
    DECLARE VARIABLE n INTEGER;
BEGIN
    templ = '';
    result = '';
    IF (s NOT LIKE '%.%') THEN
        n = 1;
    ELSE
        n = 0;
    WHILE (
    1 =
    1) DO
    BEGIN
        SELECT result
        FROM dig(:s,
            :templ)
        INTO :d;
        IF (
            d IS NULL) THEN
        BEGIN
            IF (
                result LIKE
                '.%') THEN
                result =
                    '0' ||
                    result;
            SUSPEND;
            EXIT;
        END ELSE
            IF (
```

```
                d =
                -1) THEN
                n = 1;
            ELSE
            BEGIN
                result =
                    d ||
                    result;
                IF (
                    n =
                    1) THEN
                BEGIN
                    result =
                        '.' ||
                        result;
                    n = 0;
                END
            END
        templ =
            templ ||
            '_';
    END
END
^



/*

 this reconciles strings "s1" and "s2" to make them agree with decimal point

 position.



 эта процедура приводит строки s1 и s2 к виду, в котором десятичная точка

 стоит в одном и том же разряде

*/
CREATE PROCEDURE strnormalize (
    s1 VARCHAR(1000),
    s2 VARCHAR(1000)
)
RETURNS (
    result1 VARCHAR(1000),
    result2 VARCHAR(1000)
)
AS
    DECLARE VARIABLE templ1 VARCHAR(1000);
    DECLARE VARIABLE n1 INTEGER;
    DECLARE VARIABLE templ2 VARCHAR(1000);
    DECLARE VARIABLE n2 INTEGER;
BEGIN
    templ1 = '';
    n1 = 0;
```

```
IF (
    s1 LIKE
    '%.%') THEN
    WHILE (s1 NOT LIKE
    '%.' ||
    templ1) DO
    BEGIN
        templ1 =
            templ1 ||
            '_';
        n1 =
            n1 +
            1;
    END
templ2 = '';
n2 = 0;
IF (
    s2 LIKE
    '%.%') THEN
    WHILE (s2 NOT LIKE
    '%.' ||
    templ2) DO
    BEGIN
        templ2 =
            templ2 ||
            '_';
        n2 =
            n2 +
            1;
    END
result1 = s1;
result2 = s2;
WHILE (
n1 >
n2) DO
BEGIN
    IF (
        n2 =
        0) THEN
        result2 =
            result2 ||
            '.';
    result2 =
        result2 ||
        '0';
    n2 =
        n2 +
        1;
END
WHILE (
n2 >
n1) DO
BEGIN
    IF (
        n1 =
        0) THEN
```

```
            result1 =
                result1 ||
                '.';
        result1 =
            result1 ||
            '0';
        n1 =
            n1 +
            1;
    END
    SUSPEND;
END
^

/*

 this adds string "s1" to string "s2" using old-school "column" method.



 эта процедура складывет строки s1 и s2 "в столбик".

*/
CREATE PROCEDURE stradd (
    s1 VARCHAR(1000),
    s2 VARCHAR(1000)
)
RETURNS (
    result VARCHAR(1000)
)
AS
    DECLARE VARIABLE templ VARCHAR(1000);
    DECLARE VARIABLE carry INTEGER;
    DECLARE VARIABLE d1 INTEGER;
    DECLARE VARIABLE d2 INTEGER;
BEGIN
    SELECT result1,
        result2
    FROM strnormalize(:s1,
        :s2)
    INTO :s1,
        :s2;
    templ = '';
    result = '';
    carry = 0;
    WHILE (
    1 =
    1) DO
    BEGIN
        SELECT result
        FROM dig(:s1,
            :templ)
        INTO :d1;
        SELECT result
        FROM dig(:s2,
            :templ)
```

```
        INTO :d2;
        IF (
            d1 IS NULL AND
            d2 IS NULL) THEN
        BEGIN
            IF (
                carry <>
                0) THEN
                result =
                    carry ||
                    result;
            SUSPEND;
            EXIT;
        END ELSE
        BEGIN
            IF (
                d1 =
                -1 OR
                d2 =
                -1) THEN
                result =
                    '.' ||
                    result;
            ELSE
            BEGIN
                IF (
                    d1 IS NOT NULL) THEN
                    carry =
                        carry +
                        d1;
                IF (
                    d2 IS NOT NULL) THEN
                    carry =
                        carry +
                        d2;
                result =
                    (carry - (carry / 10) * 10) ||
                    result;
                carry =
                    carry /
                    10;
            END
        END
        templ =
            templ ||
            '_';
    END
END
^

/*

 this multiplies string "s" by "dig" using repeated additions.
```

```
 эта процедура умножает строку s на целое число dig многократным сложением

*/

CREATE PROCEDURE strdigmul (
    s VARCHAR(1000),
    dig INTEGER
)
RETURNS (
    result VARCHAR(1000)
)
AS
BEGIN
    result = '0';
    WHILE (
    dig >
    0) DO
    BEGIN
        SELECT result
        FROM stradd(:result,
            :s)
        INTO :result;
        dig =
            dig -
            1;
    END
    SUSPEND;
END
^



/*

 this subtracts string "s2" from string "s1" using old-
school "column" method.



 эта процедура вычитает строку s2 из строки s1 "в столбик".

*/

CREATE PROCEDURE strsub (
    s1 VARCHAR(1000),
    s2 VARCHAR(1000)
)
RETURNS (
    result VARCHAR(1000)
)
AS
    DECLARE VARIABLE templ VARCHAR(1000);
    DECLARE VARIABLE carry INTEGER;
    DECLARE VARIABLE n INTEGER;
    DECLARE VARIABLE d1 INTEGER;
    DECLARE VARIABLE d2 INTEGER;
```

```
BEGIN
    SELECT result1,
        result2
    FROM strnormalize(:s1,
        :s2)
    INTO :s1,
        :s2;
    templ = '';
    result = '';
    carry = 0;
    WHILE (
    1 =
    1) DO
    BEGIN
        SELECT result
        FROM dig(:s1,
            :templ)
        INTO :d1;
        SELECT result
        FROM dig(:s2,
            :templ)
        INTO :d2;
        IF (
            d1 IS NULL AND
            d2 IS NULL) THEN
        BEGIN
            SUSPEND;
            EXIT;
        END ELSE
        BEGIN
            IF (
                d1 =
                -1 OR
                d2 =
                -1) THEN
                result =
                    '.' ||
                    result;
            ELSE
            BEGIN
                IF (
                    d1 IS NOT NULL) THEN
                    carry =
                        carry +
                        d1;
                IF (
                    d2 IS NOT NULL) THEN
                    carry =
                        carry -
                        d2;
                n = 0;
                WHILE (
                carry <
                0) DO
                BEGIN
                    n =
```

```
                        n +
                        1;
                    carry =
                        carry +
                        10;
                END
                result =
                    (carry - (carry / 10) * 10) ||
                    result;
                carry =
                    (carry / 10) -
                    n;
            END
        END
        templ =
            templ ||
            '_';
    END
END
^


/*


 this multiplies string "s1" by "s2" using old-school "column" method.


 эта процедура умножает строку s1 на строку s2 "в столбик".

*/

CREATE PROCEDURE strmul (
    s1 VARCHAR(1000),
    s2 VARCHAR(1000)
)
RETURNS (
    result VARCHAR(1000)
)
AS
    DECLARE VARIABLE n INTEGER;
    DECLARE VARIABLE d2 INTEGER;
    DECLARE VARIABLE templ VARCHAR(1000);
    DECLARE VARIABLE frac VARCHAR(1000);
    DECLARE VARIABLE mul VARCHAR(1000);
BEGIN
    templ = '';
    frac = '0';
    n = 0;
    result = '0';
    mul = s1;
    WHILE (
    1 =
```

```
1) DO
BEGIN
    SELECT result
    FROM dig(:s2,
        :templ)
    INTO :d2;
    IF (
        d2 IS NULL) THEN
    BEGIN
        SELECT result
        FROM stradd(:frac,
            :result)
        INTO :result;
        SUSPEND;
        EXIT;
    END
    IF (
        d2 <>
        -1) THEN
    BEGIN
        SELECT result
        FROM stradd((


                                                SELECT result
           FROM strdigmul(:mul,
            :d2)),
            :result)
        INTO :result;
        SELECT result
        FROM strdigmul(:mul,
            10)
        INTO :mul;
        n =
            n +
            1;
    END ELSE
    BEGIN
        frac = result;
        WHILE (
        n >
        0) DO
        BEGIN
            SELECT result
            FROM strshift(:frac)
            INTO :frac;
            n =
                n -
                1;
        END
        result = '0';
        mul = s1;
    END
    templ =
        templ ||
```

```
            '_';
    END
END
^



/*


 this compares strings "s1" and "s2". Unsignificant zeroes are allowed.


 эта процедура сравнивает строку s1 со строкой s2. Обрабатываются

 ситуации лидирующих/завершающих нулей

*/

CREATE PROCEDURE strcomp (
    s1 VARCHAR(1000),
    s2 VARCHAR(1000)
)
RETURNS (
    result INTEGER
)
AS
    DECLARE VARIABLE templ VARCHAR(1000);
    DECLARE VARIABLE l1 INTEGER;
    DECLARE VARIABLE l2 INTEGER;

BEGIN
    SELECT result1,
        result2
    FROM strnormalize(:s1,
        :s2)
    INTO :s1,
        :s2;

    IF (
        s1 =
        s2) THEN
        result = 0;
    ELSE
    BEGIN

        l1 = 0;
        templ = '';
        WHILE (s1 NOT LIKE templ) DO
        BEGIN
            templ =
                templ ||
                '_';
            l1 =
```

```
                l1 +
                1;
        END

        l2 = 0;
        templ = '';
        WHILE (s2 NOT LIKE templ) DO
        BEGIN
            templ =
                templ ||
                '_';
            l2 =
                l2 +
                1;
        END

        WHILE (
        l1 <
        l2) DO
        BEGIN
            s1 =
                '0' ||
                s1;
            l1 =
                l1 +
                1;
        END

        WHILE (
        l2 <
        l1) DO
        BEGIN
            s2 =
                '0' ||
                s2;
            l2 =
                l2 +
                1;
        END

        IF (
            s1 =
            s2) THEN
            result = 0;
        ELSE
            IF (
                s1 <
                s2) THEN
                result = -1;
            ELSE
                result = 1;
    END
    SUSPEND;
END
^
```

```
/*
 this calculates factorial of string "s"

 эта процедура возвращает факториал строки s
*/
CREATE PROCEDURE strfactorial (
    s VARCHAR(1000)
)
RETURNS (
    result VARCHAR(1000)
)
AS
    DECLARE VARIABLE cmp INTEGER;
BEGIN
    result = s;
    WHILE (
    1 =
    1) DO
    BEGIN
        SELECT result
        FROM strcomp(:s,
            '1')
        INTO :cmp;
        IF (
            cmp =
            0) THEN
        BEGIN
            SUSPEND;
            EXIT;
        END ELSE
        BEGIN
            SELECT result
            FROM strsub(:s,
                '1')
            INTO :s;
            SELECT result
            FROM strmul(:result,
                :s)
            INTO :result;
        END
    END
END
^


/*
 this raises string "s" to "dig" power using repeated multiplications.
```

```
 эта процедура возводит строку s в целочисленную степень методом многократно
го умножения.

*/
CREATE PROCEDURE strdigpwr (
    s VARCHAR(1000),
    dig INTEGER
)
RETURNS (
    result VARCHAR(1000)
)
AS
BEGIN
    result = '1';
    WHILE (
    dig >
    0) DO
    BEGIN
        SELECT result
        FROM strmul(:result,
            :s)
        INTO :result;
        dig =
            dig -
            1;
    END
    SUSPEND;
END
^



/*

 this divides string "s1 " by string "s2".  "Maxdigits" limits amount of

 decimal digits in resulting periodical fraction (if any).



 эта процедура делит строку s1 на строку s2. Параметр maxdigits ограничивает
количество знаков в случае,

 если результатом деления является периодическая десятичная дробь.

*/

CREATE PROCEDURE strdiv (
    s1 VARCHAR(1000),
    s2 VARCHAR(1000),
    maxdigits INTEGER
)
RETURNS (
```

```
    result VARCHAR(1000)
)
AS
    DECLARE VARIABLE n INTEGER;
    DECLARE VARIABLE cmp INTEGER;
    DECLARE VARIABLE m INTEGER;
    DECLARE VARIABLE att VARCHAR(1000);
    DECLARE VARIABLE d INTEGER;
    DECLARE VARIABLE len INTEGER;
    DECLARE VARIABLE cond INTEGER;
    DECLARE VARIABLE divisor VARCHAR(1000);

BEGIN

    divisor = s2;

    n = 0;
    cmp = 1;
    m = 1;

    WHILE (
    cmp >
    0) DO
    BEGIN
        SELECT result
        FROM strcomp(:s1,
            :s2)
        INTO :cmp;
        IF (
            cmp >
            0) THEN
        BEGIN
            SELECT result
            FROM strdigmul(:s2,
                10)
            INTO :s2;
            n =
                n +
                1;
            m =
                m *
                10;
        END ELSE
            IF (
                cmp =
                0) THEN
            BEGIN
                result = m;
                SUSPEND;
                EXIT;
            END
    END

    SELECT result
    FROM strshift(:s2)
    INTO :s2;
```

```
result = '';
cmp = 1;
len = 0;
WHILE (
1 =
1) DO
BEGIN
    len =
        len +
        1;

    IF (
        len =
        maxdigits) THEN
    BEGIN
        SUSPEND;
        EXIT;
    END

    IF (
        len =
        n +
        1) THEN
    BEGIN
        IF (
            result =
            '') THEN
            result = '0';
        result =
            result ||
            '.';
    END

    d = 9;
    WHILE (
    d >=
    0) DO
    BEGIN
        SELECT result
        FROM strdigmul(:s2,
            :d)
        INTO :att;
        SELECT result
        FROM strcomp(:s1,
            :att)
        INTO :cmp;
        IF (
            cmp <
            0) THEN
            d =
                d -
                1;
        ELSE
        BEGIN
            SELECT result
```

```
                FROM strsub(:s1,
                    :att)
                INTO :s1;

                SELECT result
                FROM strcomp(:s2,
                    :divisor)
                INTO :cond;

                result =
                    result ||
                    d;
                d = -1;
                IF (
                    cmp =
                    0 AND
                    cond =
                    0) THEN
                BEGIN
                    SUSPEND;
                    EXIT;
                END
                SELECT result
                FROM strshift(:s2)
                INTO :s2;
            END
        END
    END
END
^



/*


 ladies and gentelmen! Now here the exponent! An exponent is calulated using

 Taylor series.


 Барабанная дробь: то, ради чего всё затевалось. Экспонента рассчитывается по
 ряду Тейлора.

*/

CREATE PROCEDURE strexp (
    s VARCHAR(1000),
    maxdigits INTEGER,
    members INTEGER
)
RETURNS (
    result VARCHAR(1000)
)
```

```
AS
    DECLARE VARIABLE n INTEGER;
    DECLARE VARIABLE p VARCHAR(1000);
    DECLARE VARIABLE f VARCHAR(1000);
BEGIN
    p = '1';
    f = '1';
    n = 0;
    result = '1';
    WHILE (
    n <
    members) DO
    BEGIN
        n =
            n +
            1;
        SELECT result
        FROM strmul(:p,
            :s)
        INTO :p;
        SELECT result
        FROM strdigmul(:f,
            :n)
        INTO :f;
        SELECT result
        FROM stradd(:result,
            (
                SELECT result
                FROM strdiv(:p,
                :f,
                :maxdigits)))
        INTO :result;
    END
    SUSPEND;
END
^



/*

 we have to test it!


 неужели мы это не проверим?

*/

SELECT result
FROM strexp('1',
    10,
    20)
```

```
/*

 The result is  2.718281823. Real value is e=2.718281828, we've got an

 error in last digit. Not bad, yeah?

 Exponent opens a door to high math for us, but it is about math,

 not about Firebird.

 Thank you.

 результат: 2.718281823

 насколько я помню, e=2.718281828, то есть ошибка в последнем знаке.

 Неплохо, ведь правда?

 Имея на руках экспоненту положительных чисел, перед нами открыты все двери.

 Дальнейшее имеет отношение к учебнику математики за седьмой класс, но, увы,

 уже не имеет никакого отношения к Firebird.

 Спасибо за внимание.

*/

^
SET TERM ^^
```