```
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <ctype.h>
#define MAX_STR_LEN   255
#define PRECISION     0.000000001
#define NUM_AFTER_DOT 6
#define NOT_VALID     0
typedef char string_t[MAX_STR_LEN + 1];

char* double_to_string_v1(double num, string_t str)
{
   long long int int_part, int_dig_count, digit, mlt, pos = 0;
   double float_part;

   if(num < 0.0)
   {
      str[pos++] = '-';
      num *= -1;
   }

   int_part = (long long int) num;
   float_part = num - (double) int_part;

   if (int_part)
      int_dig_count = (long long int) log10((double) int_part) + 1;
   else
      int_dig_count = 1;

   for (long long int len = int_dig_count; len > 0; len--)
   {
      mlt = pow(10, len - 1);
      digit = int_part / mlt;
      str[pos++] = digit + 48;
      int_part %= mlt;
   }
   str[pos++] = '.';

   for (long long int i = 0; i < NUM_AFTER_DOT; i++)
   {
      float_part *= 10.0;
      digit = float_part;
      str[pos++] = digit + 48;
      float_part -= digit;
   }
   str[pos]='\0';

   return str;
}

char* double_to_string_v2(double num, string_t str)
{
```

```
    long long int num_of_digits = log10(num), digit;
    double mlt;

    while (num > PRECISION)
    {
        mlt = pow(10.0f, num_of_digits);
        digit = num / mlt;
        num -= (digit * mlt);

        *(str++) = '0' + digit;
        if (num_of_digits == 0)
            *(str++) = '.';
        num_of_digits--;
    }
    *(str) = '\0';
    return str;
}

double string_to_double(string_t str)
{
    // "^\d{1}.\d+(E\+)\d+$"
    double num = 0.0;
    int num_sign = 1, exp_pow = 0, i = 0, flag = 0;
    int dot_count = 0, neg_sign_count = 0, e_neg_sign = 0, e_count = 0, pos_sign_count = 0, e_pos_sign = 0;

    for (i = 0; str[i] != '\0' && !flag; i++)
    {
        if (str[i] == '.')
            (!dot_count) ? (dot_count = 1) : (flag = 1);

        if (str[i] == '-' && i == 0)
            (!neg_sign_count) ? (neg_sign_count = 1) : (flag = 1);

        if (str[i] == '-' && i != 0)
            (!e_neg_sign) ? (e_neg_sign = 1) : (flag = 1);

        if ((str[i] == 'E' || str[i] == 'e') && i != 0)
            (!e_count) ? (e_count = 1) : (flag = 1);

        if (str[i] == '+' && i == 0)
            (!pos_sign_count) ? (pos_sign_count = 1) : (flag = 1);

        if (str[i] == '+' && i != 0)
            (!e_pos_sign) ? (e_pos_sign = 1) : (flag = 1);

        if (e_neg_sign && e_pos_sign)
            flag = 1;

        if ((e_neg_sign && e_count == 0) || (e_pos_sign && e_count == 0))
            flag = 1;
    }
    if (flag)
```

```
    return NOT_VALID;

if (*str == '-')
{
    num_sign = -1;
    str++;
}

if (*str == '+')
    str++;

while (*str++ != '\0' && isdigit(*(str - 1)))
    num = num * 10.0 + (*(str - 1) - '0');

if (*(str - 1) == '.')
    while (*(str++) != '\0' && isdigit(*(str - 1)))
    {
        num = num * 10.0 + (*(str - 1) - '0');
        exp_pow -= 1;
    }

if (*(str - 1) == 'e' || *(str - 1) == 'E')
{
    int e_sign = 1, count = 0;
    str++;

    if (*(str - 1) == '+')
        str++;

    else if (*(str - 1) == '-')
    {
        str++;
        e_sign = -1;
    }

    while (isdigit(*(str - 1)))
    {
        count = count * 10 + (*(str - 1) - '0');
        str++;
    }
    exp_pow += count * e_sign;
}

while (exp_pow > 0)
{
    num *= 10.0;
    exp_pow--;
}

while (exp_pow < 0)
{
    num *= 0.1;
```

```
        exp_pow++;
    }

    return num * num_sign;
}

int evaluate_expression(string_t str)
{
    int sum = 0, tmp_res = 0, sign = 1, digit;

    while (*str)
    {
        digit = *str++ - '0';

        if (digit >= 0 && digit <= 9)
            tmp_res = tmp_res * 10 + digit;
        else
        {
            sum += sign * tmp_res;
            sign = (*str == '-' ? -1 : 1);
            tmp_res = 0;
        }
    }
    sum += sign * tmp_res;

    return sum;
}

int main()
{
    string_t str = "1e-2";
    printf("Input: %s\nResult: %lf", str, string_to_double(str));
    return 0;
}
```