UNITS 7 AND 8: SEMANTIC ANALYSIS and ERROR HANDLING

You are asked to write a program (using *bison*) that reads number sequences with interspersed sum operators. The numbers can be integers or can have decimals. The program should add the numbers and print the sum. In those cases where at least one of the operands is of real type, it should perform a promotion of each number or intermediate sum to the real type. If all the operands are integer, the intermediate sum and the final output should be performed using integer types.

Examples:

```
1.1 + 2
      real value = 3.100000
1 + 2
      integer value = 3
1 + 1.1
      real value = 2.100000
1.1 + 1.1
      real value = 2.200000
1.1 + 1.2 + 1
      real value = 3.300000
```

Solution:

```
%{                                      /* 1  C-bison Declaration  Section */
#include <stdio.h>
#include <string.h>

#define Tinteger  1           // definition of type integer
#define Treal    2            // definición of type real

typedef struct s_atributes {  // Define a structure for the attributes
      int value_i ;           // integer value
      double value_r ;        // real value
      int type ;              // type (integer or real)
} t_atributes ;

#define YYSTYPE t_atributes   // The stack type of bison will be an attribute register

%}

                                        /* 2  bison   Declaration Section */
%token  INTEGER
%token  REAL

%%
                                        /* 3  Syntactic - Semantic Section   */

axiom:      '\n'
          | expression
                        { if ($1.type == Tinteger) {
                             printf ("integer value = %d\n", $1.value_i) ;
                          } else {
                             printf ("real value = %lf\n", $1.value_r) ;
                          }
                        }
                  '\n' axioma;


expression:   number       { $$ = $1 ; }
          | number '+' expresion
                        { if ($1.type == Tinteger && $3.type == Tinteger) {
                              $$.type = Tinteger ;
                              $$.value_i = $1.value_i + $3.value_i ;
                          } else if ($1.type == Treal && $3.type == Treal) {
                              $$.type = Treal ;
                              $$.value_r = $1.value_r + $3.value_r ;
                          } else if ($1.type == Treal && $3.type == Tinteger) {
                              $$.type = Treal ;
                              $$.value_r = $1.value_r + (double) $3.value_i ;
                          } else if ($1.type == Tinteger && $3.type == Treal) {
                              $$.type = Treal ;
                              $$.value_r = (double) $1.value_i + $3.value_r ;
                          }
                        }
          ;

numero:      REAL         {
                             $$.value_r = $1.value_r ;
                             $$.type = Treal ;
                        }
          | INTEGER       {
                             $$.value_i = $1.value_i ;
                             $$.type = Tinteger ;
                        }
          ;
```

```
%%
                                    /* 4  C   Code  Section */

int yyerror (char *message)
{
    fprintf (stderr, "%s\n", message) ;
}


int yylex ()
{
    unsigned char c ;
    int value_i ;
    double value_r ;
    unsigned char cad [256] ;

    do {
         c = getchar () ;
    } while (c == ' ') ;

    if (c >= '0' && c <= '9') {
         ungetc (c, stdin) ;      // It is mandatory to use white spaces between
         scanf ("%s", cad) ;      // the operands and operators este scanf
         if (strchr (cad, '.') != NULL) {
              sscanf (cad, "%lf", &value_r) ;
              yylval.value_r = value_r ;
              return (REAL) ;
         } else {
              sscanf (cad, "%d", &value_i) ;
              yylval.value_i = value_i ;
              return (INTEGER) ;
         }
    }

    return c ;
}


int main ()
{
    yyparse () ;
}
```