DEPARTMENT OF COMPUTER SCIENCE
CARLOS III UNIVERSITY OF MADRID

Computer Science
Language Processors

Rules

- The duration of the test is **2.5 hours**
- Questions will not be answered during the test
- One cannot re-enter the classroom after leaving it
- The answers must be written using a pen (not a pencil)

Given the Language of lists with correctly paired parentheses, and where each list can contain sequences of integers.

Consider the following valid examples:
(((12 32 54) () (((871 3))) (34)))
(34 4 632) () (((21 65))) (42 99)
Lists with mismatched parentheses cannot valid:) ()
Lists with mismatched parentheses cannot be given:) () ([invalid]
No list can contain both numbers and other lists at the same time: (132 543 (34))

Your are asked to:

5) **Design a grammar that generates the indicated Language.**

6) **Add the needed semantic actions for the parser to process the input and return the input list by substituting each numerical sequence with the sum of its values.**

7) **Add the needed semantic actions for the parser to process the input and return, in reverse order, a sequence of products of the previously calculated sums.**

8) **What changes are needed for the grammar to avoid conflicts in bison?**

Examples:

Input		Output for 2)		Output for 3)
(((12 32 54)()(((871 3)))(34)))	➔	(((98)(0)(((874)))(34)))	➔	34 x 874 x 0 x 98
(34 4 632)()(((21 65)))(42 99)	➔	(670)(0)(((86)))(141)	➔	141 x 86 x 0 x 670

Pay attention to the provided code, which contains data structures, the function of the lexical analyzer, and other functions for dynamic memory management.
We assume that the lexical analyzer only provides the literals '(' and ')', '\n' and integers (as tokens).

To generate strings you can use strcpy and strcat functions, but we also recommend the use of sprintf, which allows a formatted printing of various types (strings, integers, etc.) on a string:

```
sprintf (temp, "%s %d", "hello", 123) ;
```

David Griol Barres, Antonio Berlanga de Jesús, Jesús García Herrero, Juan Manuel Alonso Weber

```
%{
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

typedef struct s_atributes {
      int integer ;
      char *code ;
} t_atributes ;

char temp [2048] ;
char *gen_cad () ;

#define YYSTYPE t_atributes

%}

%token  INTEGER

%%
```

GRAMMAR and SEMANTIC ROUTINES

```
%%

char *my_malloc (int nbytes)
{
    char *p ;
    static long int nb = 0;
    static int nv = 0 ;

    p = malloc (nbytes) ;
    if (p == NULL) {
         fprintf (stderr, "Error, no available memory for %d bytes \n", nbytes) ;
         fprintf (stderr, " %ld bytes have been allocated in %d calls\n", nb, nv) ;
         exit (0) ;
    }
    nb += (long) nbytes ;
    nv++ ;

    return p ;
}
```

SOLUTIONS:

1)

```
axiom:      list '\n' r_axiom
          ;

r_axiom:     // lambda
           | axiom
             ;

list:       '(' numbers ')'
             | '(' list ')'
             | list list
             ;

numbers:     // lambda
             | INTEGER numbers
             ;
```

2)

```
axiom:       list '\n'          { printf ("\n") ; }
                   r_axiom      { ; }
           ;

r_axiom:    // lambda           { ; }
           | axiom              { ; }
             ;

lista:       '('                { printf (" (") ; }
                   numbers ')'        { printf (" %d ) ", $3.integer) ; }
             | '('              { printf (" ( ") ; }
                   list ')'           { printf (" ) ") ; }
             | list list              { ; }
             ;

numbers:     // lambda          { $$.integer = 0 ; }
             | INTEGER numbers       { $$.integer=$1.integer+$2.integer; }
             ;
```

3)

```
axiom:       lista '\n'               { printf ("%s\n", $1.code) ; }
                   r_axiom      { ; }
            ;

r_axiom:       // lambda        { ; }
             | axiom                  { ; }
             ;

list:       '(' numbers ')'     {
                                     sprintf (temp, "%d", $2.integer) ;
                                     $$.code = gen_cad (temp) ;
                                    }
             | '(' list ')'           { $$ = $2 ; }
             | list list              {
                                       sprintf (temp, "%s x %s",
                                         $2.code, $1.code) ;
                                       $$.code = gen_cad (temp) ;
```

David Griol Barres, Antonio Berlanga de Jesús, Jesús García Herrero, Juan Manuel Alonso Weber

```
                                        }
            ;

numbers:      // lambda                 {$$.integer = 0 ; }
            | INTEGER numbers           {$$.integer=$1.integer+$2.integer; }
            ;
```

4) Eliminating conflicts requires the following changes:

```
axiom:      list '\n'                   { printf ("\n") ; }
                  r_axiom       { ; }
          ;

r_axiom:     // lambda          { ; }
          | axiom               { ; }
            ;

list:       '('                 { printf (" (") ; }
                  numbers ')'           { printf (" %d ) ", $3.integer) ; }
                  r_list                { ; }
            | '('               { printf (" ( ") ; }
                  list ')'              { printf (" ) ") ; }
                  r_list        { ; }
            ;

r_list:      // lambda          { ; }
            | lista                     { ; }
            ;

numbers:     // lambda          { $$.integer = 0 ; }
            | INTEGER numbers           { $$.integer=$1.integer+$2.integer; }
            ;
```