TDT 4205
Problem Set 3

The deadline for this problem set is friday, October 12th. Submissions will only be accepted through *It's learning*. This is a larger programming exercise, worth twice as much as the previous assignments.

The full grammar of VSL (Very Simple Language) is as follows:

program → function_list
function_list → function | function_list function
statement_list → statement | statement_list statement
print_list → print_item | print_list ', ' print_item
expression_list → expression | expression_list ', ' expression
variable_list → variable | variable_list ', ' variable
argument_list → expression_list | ϵ
parameter_list → variable_list | ϵ
declaration_list → declaration_list declaration | ϵ
function → FUNC variable '(' parameter_list ')' statement

statement→ assignment_statement | return_statement | print_statement
| null_statement | if_statement | while_statement | block

block → '{' declaration_list statement_list '}'
assignment_statement → variable ':' '=' expression
return_statement → RETURN expression
print_statement → PRINT print_list
null_statement → CONTINUE
if_statement → IF expression THEN statement FI
| IF expression THEN statement ELSE statement FI
while_statement → WHILE expression DO statement DONE

expression → expression '+' expression | expression '-' expression |
expression '*' expression | expression '/' expression | '-' expression |
'(' expression ')' | integer | variable | variable '(' argument_list ')'

declaration → VAR variable_list
variable → IDENTIFIER
integer → INTEGER
print_item → expression | text
text → TEXT

The IDENTIFIER token refers to a case sensitive string of digits, letters and underscore which begins with a letter.

1 Construction of syntax trees

The basic structure of a syntax tree node contains the type of node (enumerated type nodetype_t, provided), its number of children, a table of pointers to those children, and an optional label 'data' (which should be set to NULL when unused):

```
typedef struct n {
    nodetype_t type;          // Type of this node
    int n_children;           // Number of children
    struct n **children;      // Pointers to child nodes
    void *data;               // Data label for terminals and expressions
} node_t;
```

1.1 10%

Create a function
`node_t *allocate_node (nodetype_t type, int n_children, void *data)`
which creates a labelled node with space for n children, without assigning any links to them.

1.2 10%

Create a function
`void destroy_node ( node_t *discard )`
which frees the memory associated with a node, but does not affect its children.

1.3 10%

Create a function
`node_t *create_node (nodetype_t type, int n_children, void *data, ...)`
which creates a labelled node as a parent of the n child nodes given in the variable-length argument list.

1.4 10%

Create a function
`node_t *destroy_subtree ( node_t *root )`
which recursively frees the memory associated with the subtree beginning at *root*.

1.5 10%

Create a function
`void print_node ( FILE *output, node_t *root, int indent )`
which recursively prints a textual representation of the subtree beginning at *root*, beginning each line by indenting the text by a number of spaces which is equal to the current depth in the tree.

1.6 20%

Create a scanner/parser pair which construct the full syntax tree of a VSL program.

2 Simplification of syntax trees

2.1 30%

Create a function
`node_t *simplify_tree ( node_t *root )`
which modifies a syntax (sub)tree as follows:

- remove single-child nodes *statement, argument_list, parameter_list*
- flatten list structure subtrees into a single list node with n children
- simplify expression subtrees which only depend on constant values into a single INTEGER node

Note that although some of these simplifications can be made directly in the semantic actions of the parser (for slightly greater efficiency), you are requested to keep the construction and simplification of the tree separate. This is done in order to make it easier to test the correctness of the two phases separately; we will not be parsing any programs of such size that speed becomes a concern anyway.