TDT4205 Problem Set 4

Answers are to be submitted via Blackboard by March 20^{th}.
This assignment will account for 10% of your final mark.

1 Three-Address Code (TAC) (40%)

The following VSL program calculates the n^{th} Catalan number,

$$C_n = \frac{2n!}{(n+1)!n!}$$

Translate the program into TAC. You may assume that printing can be done via a call to an external function with an argument signature of your own choosing. *(Feel free to make further assumptions about the execution environment if you find it necessary, but state them in your answer.)*

```
func catalan( n )
begin
    print factorial(2*n) / (factorial(n+1)*factorial(n))
    return 0
end

func factorial( n )
begin
    var i, result
    result := 1
    i := 1
    while i < (n+1) do
    begin
        result *= i
        i += 1
    end
    return result
end
```

2 Symbol Table Creation (60%)

2.1 Initialization (15%)

After creating and simplifying the syntax tree, the task now is to create a symbol table. Implement `create_symbol_table` to initialize `global_names` and call the subsequent functions to fill it. The function body contains some code demonstrating the usage of `tlhash`, remember to remove this when starting your own implementation.

2.2 Finding globals (20%)

Implement `find_globals`. This function should create symbol table entries for global variables and functions. Every function will need their own, local symbol table. This can be created now and initialized with the function parameters

2.3 Finding local symbols and strings (20%)

Implement `bind_names` to traverse each function, adding entries for new local variables and binding variable references to the appropriate table entries. Additionally, add string literals to the `string_list` array and replace the data field in the string node with its index in `string_list`. When generating code we would like to write out all string literals at once as static variables.

2.4 Printing and cleanup (5%)

Implement `print_symbol_table` to display your symbol table. This function is called in the main program if you pass the `-u` flag on execution. Finally, implement `destroy_symbol_table` to free the resources you have allocated.