UCT 2006 CSC3005 Compilers

Practical Assignment 1: Symbol Tables

Build a **functional hash-table** implementation of a Symbol Table to handle simple constant definitions and uses in a statically-scoped block structured language.

Use a stack to maintain copies of hash table arrays for outer scopes. Use a hash table array size of 5, with chaining for collision resolution. Use any reasonable hashing algorithm.

Your main program must accept a single command-line parameter that is the name of a data file containing actions to be performed on the symbol table. The following is a sample of the data file contents:

```
beginscope
define a 1
use a
endscope
use a
```

Your program must scan through the data file in sequence and perform the necessary operations on the symbol table.

Your output must be identical to the input, except that each "use" line must have the value of the constant indicated. For example, the above input file will result in the following output, printed to stdout:

```
beginscope
define a 1
use a = 1
endscope
use a = undefined
```

You may use Java or any other programming language that the TA agrees to. You may use the built-in hash table if there is one, but make sure that for your purposes you only replicate the array and not the entire hash table on each beginScope.

Test your code with and provide output for the 3 data files attached (testdata.tar.gz). In each case, use output redirection to capture the output and save it to appropriately-named files (e.g., test1.out).

Your assignment will be marked according to the following criteria:
Correctness (45%) (test programs generate correct output at each layer)
Documentation (15%) (comments within hand-written source – no report necessary)
Efficiency (10%)
Stress (30%) (hidden test programs generate correct output)