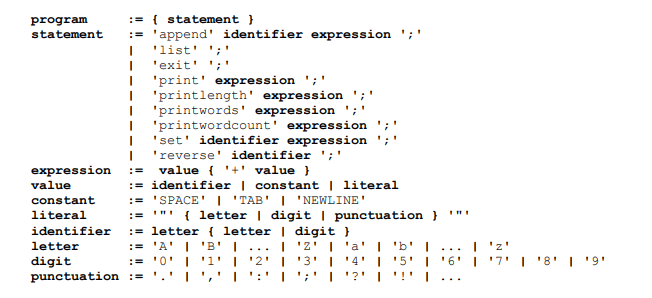
|  |  |
| --- | --- |
| A picture of a winding road and trees  String interpreter  A Haskell project | Abstract  Assignment One for course 159.341  Damien DeCourcy  19042551 |

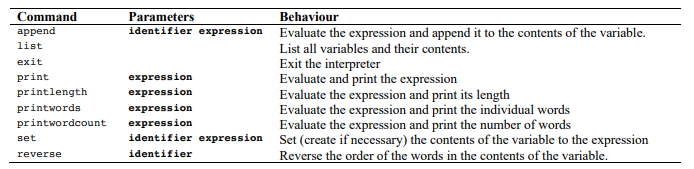
# Introduction

The interpreter is designed based on the brief provided to us.

The language specified is as follows:



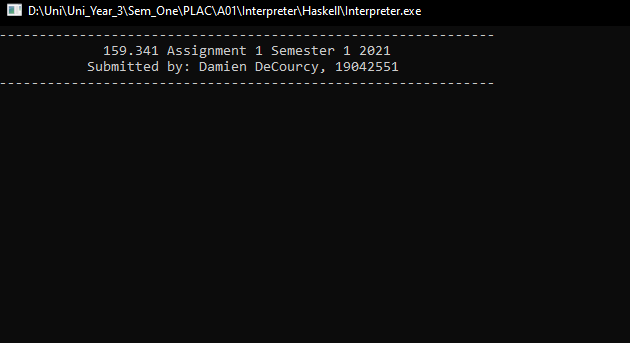
The command set is as follows:



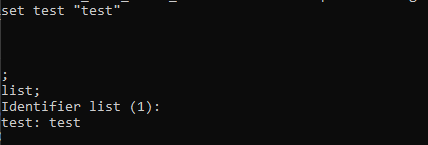
This document hopes to outline the basics of the interpreter.

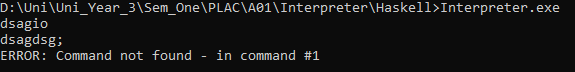
# Running the interpreter in console mode

When running the interpreter in console mode, no input prompt is provided (as shown below):



All commands must be terminated with a ‘;’ before pressing the enter key or it will result in a continual parsing of white space until one is entered (or potentially an error):



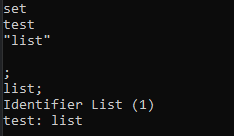


Commands can be strung together without line breaks…



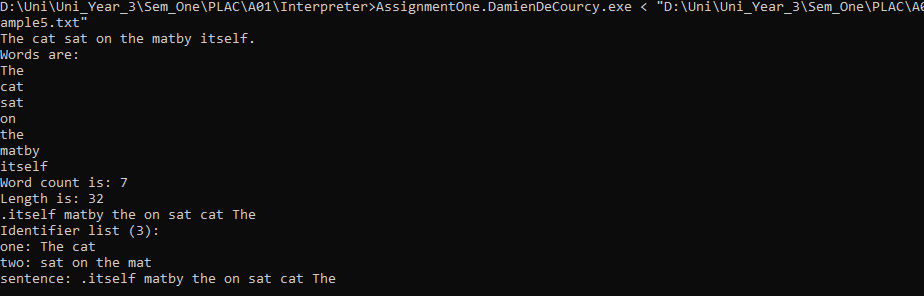
A carriage return will not execute the line of code entered:





# Running the interpreter with input redirect

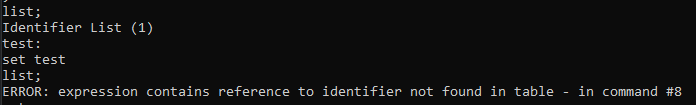
This will parse a text file until either an exit command or EOF character is encountered:



You will notice my word count is different to the “expected” output in the brief. I will cover this in the “caveats” section.

# Error Handling

When handling an error a cmd # of the command in which the error occurred will be returned:



When reading from a file this number may not correlate to the line number of the command. This is due to commands spanning multiple lines, but being parsed as one singular command.

# Escapable Characters

There are no escapable characters in this parser. This means double quotes are not accepted in strings, and will always terminate a literal string.

# Caveats

## Word count of variables

### Difference due to storing of word

I have stored all variables as strings. However, in the “possible outputs” the variables appear to be stored as tokenized strings. Subsequently my ***append***command appends behaves differently to the example append command.

One example; when conducting a “word count”, I read the string and break on SPACE characters. The example appears to do this, but with each token in the string. This means that my word count can differ from the “possible outputs” examples.

**Example code:**

set test “test”;

append test “more test here”;

**My interpreter:**

test: “testmore test here”

word count: 3

**Example interpreter**

test: “test” ”more test here”

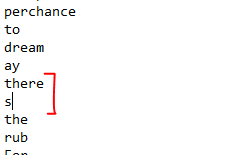
wordcount: 4

This was not expressly stated how we should treat strings (in the brief) and I only picked it up way too late to change it.

### Difference due to split on Non Grammar symbols

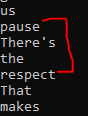
In example output 4 I noticed the words were split on symbols, causing words such as ‘There’s’ to be produced as “There” and “s”

**Example Output**



**My output**

I do not do this, and this also effects the word count:



## Literals

In my parser, literally any symbol can be included in the literal, the only exception is the non-escapable “ character as described above.

## Grammar

The following symbols are accepted as punctuation and will terminate a word in word count/length functions:

**. , : ; ! ? ‘ “**

Any other symbol is up to the user to figure out. Word and (English) language integrity are the user’s responsibility.