1 Problem Description

Problem 2 involves creating a lexical analyzer for Fortran using flex, given the specifications in the assignment PDF and on Piazza.

2 Instructions to Run

The subdirectory 'problem2' contains 3 files: prob2.l, prob2.sh and this PDF. To run, simply:

- Open the file prob2.sh. It contains a variable named 'file_name'. Rename it to the relative path of the testcase file. Note that the variable name should **not** contain the file extension as it is always assumed to be .f08. For example, if the testcase file is private1.f08, simply rename the variable file_name to private1. The script with handle the rest.
- Now, run prob2.sh. It can be executed in 2 ways:
  - ./prob2.sh -f: This redirects the output to the file {file_name}.output
  - ./prob2.sh: This will simply display the output to stdout.
- The script automatically deletes the extra files that it has created (a.out and lex.yy.c).

3 Corner/Error Cases and Format of Output

- The scanner scans all the tokens till it encounters EOF and reports all (if any) errors it has encountered.
- The output is sorted by lexeme.
- All tokens except strings are case-insensitive. Thus, all appearances of a given token are added to the common counter. However, as elaborated on Piazza, they are enlisted in different rows in the output.
- The following are error cases that are flagged by the scanner:
  - **Invalid strings**: Any unclosed string/having occurrences of ' in between are reported.
  - **Invalid identifier**: Identifiers must begin with a letter. Any identifier beginning with an _ (underscore) or a digit is reported as an error.
  - **Longer names**: Fortran only allows names of up to 63 character length. Any name longer than 63 characters is reported as an error.
  - **Unrecognized characters**: Characters that are not accepted by the language are marked by the lexical analyzer.
  - **Floating point exponents**: The exponent can only be a digit string. Thus, cases like 1.2E5.5 are flagged as an error.
- The following cases, in my opinion, were not very direct and required some thinking/modification in the code to incorporate and should be tested:
  - Comment in the last line of the file: my initial flex specification mentioned that a comment is ! and terminated with a newline character (
    n). However, if the comment is placed on the last line, that resulted in an error: the terminating newline character is thus, optional.
  - Strings enclosed in " " with multiple occurrances of ' in it: For example,
    "This is an 'invalid' string"
    This entire string should be marked as an error.