CS335 - Compiler Design - Assignment 1

Harsh Bihany (210406)

Problem 1

The following entails the scanner features and specifications for the language “Kanpur”.

1.1 Execution Details

There are two modes of execution

1. That prints the errors (if any) in the code.

   `./prob1.sh <relative-path-.knp> -y`

   for example:

   `./prob1.sh ./testcases/public1.knp -y`

2. That does not print the errors and only the maximum ‘matchable’ tokens.

   `./prob1.sh <relative-path-.knp>`

Note. *After extracting the zip into the directory, there might be a need to make the attached script executable. Kindly* `chmod` *the bash script accordingly. Also there is a need of* `g++ --std=c++17` *or above, which is generally available. Because the generated* `C++` *code uses some specificities from those versions which were absent in older versions. If the script runs without error on the* `public1.knp` *file, everything is golden!*

1.2 Error Specification

1.2.1 Valid Identifiers

If any identifier begins with a digit, the scanner should report an error (if the `-y` option is used) and ignores the inclusion of the associated characters and the possible tokens they might form in the generated table.

For example, consider a statement like `12X := 34`, the scanner reports an error:

```
line no.: <no.>          An identifier name starts with a letter.
```

`<no.>` is replaced by the appropriate line number.

1.2.2 Valid Strings

For ill-formed strings: Unclosed strings OR strings that start with a double quote and have a single quote within them and vice versa, the scanner should report an error.

For example, consider a string `"x is 'greater than y"`, the scanner should report an error:

```
line no.: <no.>          Errorneous string `"x is 'greater than y"`.
```

1.2.3 Valid Numeric Literals

Errors in numeric literals reported by the scanner belong to one of the following category:

1. Floating point used in hexadecimal representation.
2. Leading zeroes, both in the case of decimal and hexadecimal notation.
3. Floating point numbers with more than 6 decimal digits.
4. A floating point number without a characteristic. Eg. number of the form .89.

For example, consider a string `y := 20.7890825;`, the scanner should report an error:

```
line no.: <no.>          No more than 6 decimal digits allowed.
```

and other specific errors are reported.

1.2.4 Other errors

1. For unrecognized characters, the scanner throws an error.
2. *Special*: If an assignment operator = is used, a specialized error is thrown

```
line no.: <no.>          Did you mean EQL or :=?
```

Problem 2

The following entails the scanner features and specifications for the language "Fortran 2008".

2.1 Execution details

Largely the same as the previous case, except the script is nor `prob2.sh`.

2.2 Error Specification

2.2.1 Valid Identifiers/Names

If a name begins with an underscore/digit or is more that 63 characters long, the scanner throws specialized errors in both cases.

For example, consider a name `_Compilers`, the scanner should throw an error:

```
line no.: <no.>          An identifier name starts with a letter.
```

and likewise for the other type of error.

2.2.2 Valid Numeric Literals - INT / REAL

A specific error treated by the scanner is when the exponent value is not a signed integer but a real entity. For example.: `x = +5.e-12.34`, the scanner should report an error:

```
line no.: <no.>          Exponent can only be a signed integer literal.
```

2.2.3 Valid Char Literals/Strings

For ill-formed strings: Unclosed strings OR strings that start with a double quote and have a single quote within them, the scanner should report an error. The errors reported are similar to the previous Problem 1 case as discussed above.
