

**CMSC430 Scanner – Version 3**  
**Project 1 – Due 11:59pm February 20, 2007**

## 1. Scanner Specifications

Using either flex or Jflex, write a scanner for NIP. Your scanner should have 3 major sections: (1) Token generator, (2) Program lister, and (3) Symbol table. Each will be described separately.

### 1.1. Token generator

**Format of tokens:** Each token will be a pair of integers. The first member of the pair will be the *class* of the token, while the second member will be the *member* of that class. Token classes are as follows:

- Reserved word, which is the list on page 3 of the NIP language definition (e.g., START=1, FUNCTION=2, PROCEDURE=3, ..., RETURN=16). (See %token command in Lex.) *Member* is not needed here.
- ID – Identifier. *member* will be the location of that identifier in the symbol table.
- CONSTANT – integer constant. [*Change in version 3: Delete this requirement: The value of the constant will be stored in a symbol table entry.*] *member* will be the value of that symbol table entry.
- LITERAL – string constant. *member* will be the symbol table entry containing the address of the LITERAL. The LITERAL itself will be in some predefined CHAR array. The array contents, starting at that location will be the size of the literal followed by its value, without quotes.

For example, the two literals "my constant" and "next case" will be stored as:

```

1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22
11 m y      c o n s t a n t 9 n e x t      c a s e
with the corresponding tokens being: (LITERAL,1) and (LITERAL,13).
```

- Operators. These will be a series of token types: ADDOP, MULTOP, and the symbols (, ), :, ;, ,, :=. *member* will be used to distinguish between + and - and between \* and /.
- BRACKETS – Used for { and }. { = LEFTBRACKET and } = RIGHTBRACKET
- ERROR— error token. *member* is any implementation-defined value.

If the scanner finds an error, print a message, return class error, and then begin with the next symbol the next time the scanner is invoked.

Comments and blanks (not within literals) are ignored by the scanner.

## 1.2. Program lister

If the *s* option is specified, each line of the source program shall be listed on the output file as soon as it is read in. Each NIP implementation may include additional information in the listing.

Error messages shall appear after the line containing the NIP statement with the error. Error messages shall have the syntax:

**\*\*\*ERROR\*\*\*** *text*

where *text* is an appropriate implementation defined message.

## 1.3. Symbol table

For the scanner phase, a simple symbol table shall be used. You will replace it later by a more complete symbol table and semantics routine. The first identifier seen by the scanner shall have number 1 as its member number, the second shall have 2, etc.

- Ignore scope rules for now. Keep this simple, since it will be replaced later by a more complex symbol table routine. For example, if the first three identifiers seen by the scanner are: ProgramName, NewName and ProgramName, the three tokens shall be: (ID,1), (ID,2), and (ID,1).

## 2. Project requirements

- Build a scanner using flex or Jflex according to the above specifications.
- Call your scanner from a main program such that after each call the next token is returned.
- Write out each token after each call to your scanner. Thus your output should be:

```

First line of source (if s option specified)
token 1 for line
token 2 for line
...
last token for line
Next line of source (if s option specified)
token 1 for line
...

```

- Some test data will be available after February 15, 2007 in the files *~mvz/test0.nip* through *test4.nip* on the grace cluster. *test0.nip* and *test4.nip* are the most informative. These test files can be used for all phases of the compiler project.
- Projects are due February 20, 2007 before 11:59pm. Turn in a well documented source program, a make file to compile it, and a file of test data that you believe tests the scanner to the submit server. Your scanner will be tested with this test data as well as other test data. There is a penalty for submitting your project late. 10% a day for up to 3 days. After that the penalty goes up even more. (Amount TBD).