Creating a Compiler

COMP360

Compiler Project

- You will create a compiler
- There are three assignments
  - Create a lexical scanner to read the source code and create a list of tokens
  - Create a parser that inputs the list of tokens and determines if the language has the proper syntax
  - Modify the parser to create output to execute the program

Team Project

- You may create teams of two students or you may do the assignments by yourself
- You can change teams after each phase

Languages

- Your compiler must compile one of the following languages
- Snowflake – a subset of Snobol
- Musicol – A language that defines music
- Language of your creation *(must be approved)*

Defining Your Own Language

- The language must be context free
- There must be more than one program you can reasonable write in the language
- There must be an output

Lexical Scanner

- Due by noon on Monday, March 21, 2016
- You must specify which language your scanner will process
- You can write your scanner in any language you like (i.e. C++, Java, Python, etc.)
- You may use scanning tools if your wish (i.e. lex or flex)

Parser

- Due by noon on Wednesday, March 30, 2016
- You must specify which language your parser will process
- You can write your parser in any language you like (i.e. C++, Java, Python, etc.)
- You may use parsing tools if your wish (i.e. Yacc or bison)

Intermediate Code Generation

- Due noon on Wednesday, April 6, 2016
- Our compilers do not have to create machine language
- Your compiler can create output in a high level language such as C++ or Java
- The generated program must compiler and run correctly
- When executed, the generated program must do what was written in the defined language

The Snowflake Language

- Snowflake is a subset of Snobol, an old text processing language
- The only variable type in Snowflake is string
- Pattern matching is a major part of Snowflake
- Patterns can be a string, a sequence of string or alternative strings

```
cow = 'The cat in the hat'
cow 'cat' = 'dog'
output = cow  // displays "The dog in the hat"
```

Snowflake BNF

program → labelline | labelline **EOL** program

labelline → **~name** line | line

line → asg | asg **:** nextline

nextline → **(name)** | **s(name)** | **f(name)** | **s(name)f(name)**

asg → **name =** equation | **name** pattern **=** equation | **name** pattern

equation → varconst | varconst equation

pattern → alt | alt **!** pattern

alt → saveitem alt | saveitem

saveitem → item | item **$ name**

item → **(** pattern **)** | varconst | **break(** varconst **) | span(** varconst **)**

varconst → **name** | **constant**

Musicol

- Notes are represented by the inverse of their length followed by the note

```
pattern mary {time 4/4 4C4 8B3 4G3 4B3}
pattern lamb {4C4 4C4 2C4}
pattern snow {4C4 4C4 4C4 4C4}
play 1 times[ mary trans lamb{0 -1 0} mary]
play 2 times[snow]
play 1 times[2G3]
```

Musicol BNF

song → patternList playlist
patternList → pattern | pattern patternList
pattern → **pattern name {** notepat **}**
notepat → note | **name**
note → **number** freq | **number** (freqList)
freq → **letter number** | **letter number b** | **letter number #**
freqList → freq | freq | freqList
trans → **trans name {** signednumList **}**
signednumList → signednum | signednum signednumList
signednum → **+number** | **-number** | **0**
playlist → play | play playlist
play → **play number times [** stuff **]**
stuff → name | trans | note