Take Home Guidelines

1. Import the following Racket libraries to help with the parser implementation

```
#lang racket
(require parser-tools/yacc
      parser-tools/lex
      (prefix-in : parser-tools/lex-sre)
      syntax/readerr)
```

2. List the empty tokens

```
(define-empty-tokens op-tokens (newline EOF))
```

3. List the token categories including (e.g. DELIMITER, OPERATOR, etc.)

```
(define-tokens value-tokens ( LIST HERE) )
```

4. Use the following construction to define the lexical rules (token rules)

```
(define lex (lexer [ rule1 ] [rule 2  ]  ….. [rule n ] ))
```

5. The lexical rules (token rules) can be written for example as follows:

```
["then” 'THEN]
[(:or "true" "false")  'BOOL]
```

6. Define now the parse using the construction

```
(define cicom-parser (parser ( grammar ( THE GRAMMAR RULES HERE))))
```

7. The grammar syntax can be easily defined. For example

```
(exp ((term binop exp) "ACTION")
          ((term) "ACTION")
          ((IF exp THEN exp ELSE exp) "ACTION")
          ((LET def IN exp) "ACTION")
          ((MAP idlist TO exp) "ACTION"))
```

8. Use the following construction for testing

```
(define file (file->string "NAME OF FILE"))
(define (lex-this lexer input) (lambda () (lexer input)))
(cicom-parser (lex-this lex (open-input-string file)))
```

9. To define Def+ you may try a construction as follows

```
(def ((ID ASSIGN exp SEMICOLON def) "ACTION")
```