**Name: Milson Munakami**

**Student ID: 114012811**

**Course: CS 354 Programming Languages**

**Instructor: Dr. Jim Buffenbarger**

**Assignment 1:**

TA1:

Exercise 1.1

I am using Java as the imperative programming language to demonstrate these errors type:

1. Lexical error: int sum+; because sum+ is invalid identifier as + is not applicable to char identifier
2. Syntax error: missing “}” at the end of a Class.
3. Static semantic error: accessing undeclared method
4. Dynamic semantic error: accessing an element out of bound of an array
5. Error that can't reasonably by caught: using same method name as a variable inside the method

Exercise 1.6

Make utility needs to perform all necessary recompilations by check the makefile data base and the last-modification times of all files in order to decide which of the files need to be recompiled.  This dependency management utility tend to cause increase the size of the source files. With size its performance will degrade. Also maintaining the prerequisite files, linking steps and declaring all the object files to look for small or no evident change can make it slow. The order of creation of any program can fail it. There might be circular dependency between files which will show warnings. It seems unnecessary to specify the order of build process to make it run successfully. IT seems make‘s reliability is poor, especially for larger or incremental builds.

Exercise 2.1

Regular expression

*letter* → a|b|c|…|z|A|B|C|…|Z|

^( *letter*?:( *letter*? **not** time|hour|minute).)\*$

Note: Here **not** is supposed to be an operator.

Exercise 2.13

Analysis a CFG

1. Parse Tree : ***foo(a,b)***

stmt

subr\_call

Id (foo)

)

arg\_list

(

expr

args\_tail

expr\_tail

,

primary

Id(a) 

arg\_list

expr

args\_tail

ℇ

ℇ

expr\_tail

primary

ℇ

Id(b) 

b.Canonical (Rightmost) Derivation

program =>stmt

=>subr\_call

=>id ( arg\_list )

=>id ( expr args\_tail )

=>id ( expr , arg\_list )

=>id ( expr , expr args\_tail )

=>id ( expr , expr ℇ )

=>id ( expr , primary expr\_tail )

=>id ( expr , id(b) ℇ )

=>id ( primary expr\_tail , id(b) )

=>id ( primary expr\_tail , id(b) )

=>id (id(a) ℇ , id(b) ℇ )

=>id(foo) ( id(a) , id(b) )

Exercise 2.17

Modify a CFG to allow if or while statements

1. program 🡪 stmt\_list $$
2. stmt\_list 🡪 stmt\_list stmt
3. stmt\_list 🡪 stmt
4. stmt 🡪 id:= expr
5. stmt 🡪 read id
6. stmt 🡪 write expr
7. **stmt 🡪 if cond then stmt fi**
8. **stmt 🡪 while cond do**
9. **stmt 🡪 od**
10. **cond 🡪 expr**
11. **cond 🡪 cond cond\_op expr**
12. expr 🡪 term
13. expr 🡪 expr add\_op term
14. term 🡪 factor
15. term 🡪 expr mult\_op factor
16. factor 🡪 ( expr )
17. factor 🡪 id
18. factor 🡪 number
19. **cond\_op 🡪 <**
20. **cond\_op 🡪 >**
21. **cond\_op 🡪 ==**
22. **cond\_op 🡪 !=**
23. **cond\_op 🡪 <=**
24. **cond\_op 🡪 >=**
25. add\_op 🡪 +
26. add\_op 🡪 -
27. mult\_op 🡪 \*
28. mult\_op 🡪 /