```
****************************************************************
* Andre Maldonado
* CS354: TA1-1
* 2 October 2019
****************************************************************
```

**question 1.1 is using Java

1.1)

a)Lexical Error:

```
public class lex
{
    inte x = 4;
}
```

b) Syntax Error:

```
public class syn
{
    int a = 3;
    int b = 5
    if( a <= b){
          a++
}
```

c) Static Semantics Error:

```
public class ssem
{
    x=4
  System.out.println(x);
}
```

d) Dynamic Semantics Error:

```
public class dsem
{
    int[] array = new int[4]'
    System.out.println( array[4] );
}
```

e)

```
public void foo()
{
    int a = 0;

  while(false)
  {
      a++;
      System.out.println("Unreachable");
  }
}
```

1.8) This dependence is mildly accurate. If there are changes made to the code in file B, then obviously file A should also be recompiled. If the changes to file B do not affect the code, then there should be no need to recompile A. Some examples could be if comments or documentation are either added or changed in file B. These things do no affect the functionality of the file so there should not be a need to recompile file A. If the dependency only goes one way, then any changes to file A will not cause file B to be recompiled.

2.1)

```
a) "(H|e|l|l|o|+$)*\*(h|e|l|l|o|"|)"

b) (* (+) * * ) + {(+) *}

c) C_constant -> int_const|fp_const
   int_const -> ( oct_int|dec_int|hex_int)int_suffix
   oct_int -> 0 oct_digit*
   dec_int -> nonzero_digitdec_digit*
   hex_int -> (0x |0X )hex_digithex_digit*
   oct_digit -> 0|1|2|3|4|5|6|7
   nonzero_digit -> 1|2|3|4|5|6|7|8|9
   dec_digit -> 0 |nonzero_digit
   hex_digit -> dec_digit | a| b| c| d| e| f| A| B| C| D| E| F
   int_suffix -> |u_suffix(l_suffix| ll_suffixe| )
   |l_suffix(u_suffix|  ) | ll_suffixe( u_suffix|  )
        u_suffix -> u | U
        l_suffix -> l | L
        ll_suffix -> ll| LL


d)     Ada_int ->digit ((_| ?)digit)*
       extended_digit ->digit |a|b|c|d|e|f|A|B|C|D|E|F
```

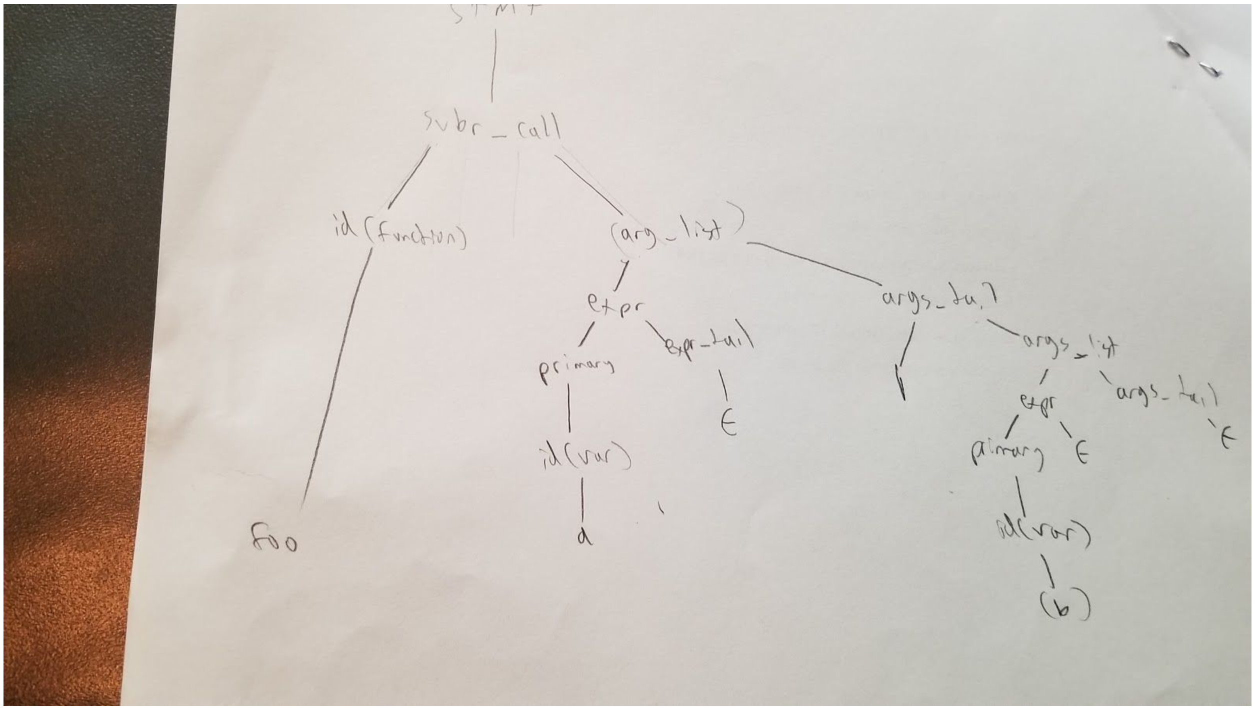
Ada_extended_int -> extended digit ((_| ? extended digit)*
Ada_FP_num ->((Ada_int((.Ada_int| ?))
|(Ada_int# Ada_extended_int ((.Ada_extended_int)| ?) #))
(((e |E)(+ | - | ?)Ada_int) | ?

e) digit+ # * (. # * |) | digit* .digit+ # *

f) nonzerodigit -> 1|2|3|4|5|6|7|8|9
digit -> 0| nonzerodigit
number -> $* *(0|nonzerodigit(digit| digit digit)(| .digit digit)

2.13)

a)



b)
id (arg_list
foo expr arg_tail
foo primary expr_tail arg_list
foo id ∈ expr arg_tail
foo a, primary expr_tail ∈
foo a , id ∈
foo a, b

2.17)

program -> stmt_list $
stmt_list ->stmt_list stmt
stmt_list ->stmt

```
stmt -> id:=expr
stmt-> if (expr) then stmt_list if
stmt->while (expr) do stmt_list
stmt-> read id
stmt->write expr
expr ->term
expr -> expr add_op term
expr-> expr cond expr
term->factor
term-> term mult_op factor
factor->id(expr)
factor->num
add_add -> +
add_sub -> -
mult_mult -> *
mult_div-> /
Cond -> >
cond-> <
cond-> <=
cond-> >=
Cond -> !=
Cond -> ==
```