UTD

Karan Sahu
KXS 190007
CS 4337.503

CS/CE 4337 - Assignment#2
Due Date: 10/2/20, 11:59 pm

1-Write an evaluation for Java, using the criteria described in chapter 1. (at least 10 lines)

2-Is the following grammar ambiguous? Why ?
<S> → <A> a <B> b
<A> → <A> b | b
<B> → a <B> | a

3-Write a denotational semantics mapping function for java while loop
4-Compute the weakest precondition for the following statements
A.

```
x = 4 * (2 * x+ y) ;
y = 4 * x - 1 {y > 7}
```

B.

```
if (x == y)  y=3*x+ 3
else y= 2 * x;  {y > 2}
```

5-Write an EBNF rules that describes the following while statement of Java. Then, write the recursive-descent subprogram in Java for the EBNF rule. Please summit your source code and a screen shot of the parsing of the following examples.

```
do {
  if ( number % 2 == 0 )
     even ++;
   number=number+1;
}
while (number <= 10)
```

6-Given the following grammar and the right sentential form, draw a parse tree and show the phrases and simple phrases, as well as the handle.
Grammar: S→XyY|yXz X→Xy|xYY Y→Xz|zYy|z
right sentential form: xXzzzyyz

7-Show a complete parse, including the parse stack contents, input string, and action for the string (id)+(id*id), using the grammar and parse table in Section 4.5.3.

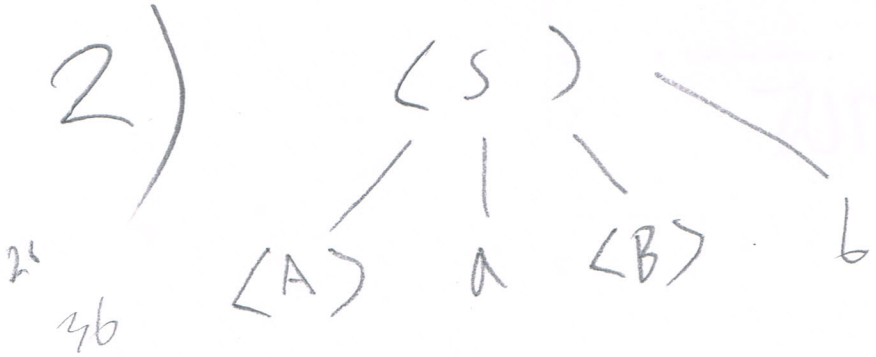
1) 1. Readability - Obviosly there are different variables effecting how easily a user can read a Language. such as previous experiance. But overall I would argue Java has good & bad readability depending on how it was written. It has very high expsessivity which can be bad for readability but also is similar to other Languages like C# and python making it an easy language to read for a variety of programers.

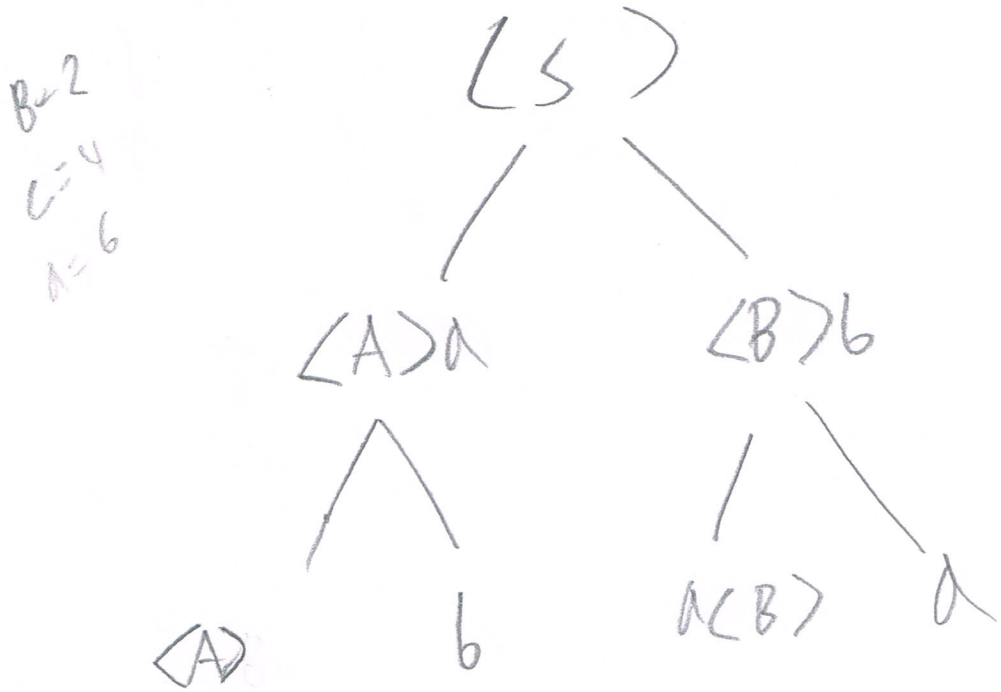
2. writability.- I think is has good writability everything you want to do can be don straight forward and looked up but again the expresivity can effect writability.

3. Reliability - since Java has been around for so long and has so many users its rare for it to have problems or errors to go past compiler

4. Wtamate cost.- relative to other programing Languages I would argue a low Wtamte Cost because it is so common & well known Language and there is a reason for that clearly it is benificiN and efficient for many companys.

2)





b* λ a* b

This grammar is not ambiguous b/c there is only 1 way you draw the parse tree for any given string

3)

M_1(while B do L, S) Δ = if M_b (B, S) == undef

then error

else if M_b (B, S) == false

then S

else if M_{S1} (L, S) == error

then error

else M_1(while B do L, M_{S1}(L, S))

4) $X = 4 * (2 * X + Y)$;

A) $Y = 4 * X - 1 \quad \{ Y > 7 \}$

$7 < 4 * X - 1$

$X > 2$

$2 < 4 (2X + Y)$

$2 < 8X + 4Y$

$2 - 8X \leq 4Y$

$4Y > 2 - 8X$

$Y > \frac{1}{2} - 2X$ ← weakest cond

$7 < 4x - 1$

$8 < 4x$

$2 < x$

$x = 8x + 4y$

$-7x = 4y$

$x = -\frac{4}{7}y$

$2 < -\frac{4}{7}y$

$\frac{14}{-4} > y$

$-\frac{7}{2} > y$

$y = -4$

46)

$2 < 2 \cdot x$

$1 < x$

$x > 1$

$2 < 3 \cdot x + 3$

$-1 < 3x$

$-\frac{1}{3} < x$

$x > -\frac{1}{3}$

$x > 1$

weakest condition

5) <doWhile> -> do { <while cond true> } while (<cond>)

<assign> -> id ((= <expr> | ++ | --))

<cond> -> <expr> (> | < | <= | >= | != | ==) <expr>

<expr> -> <term> { (+ | -) <term> }

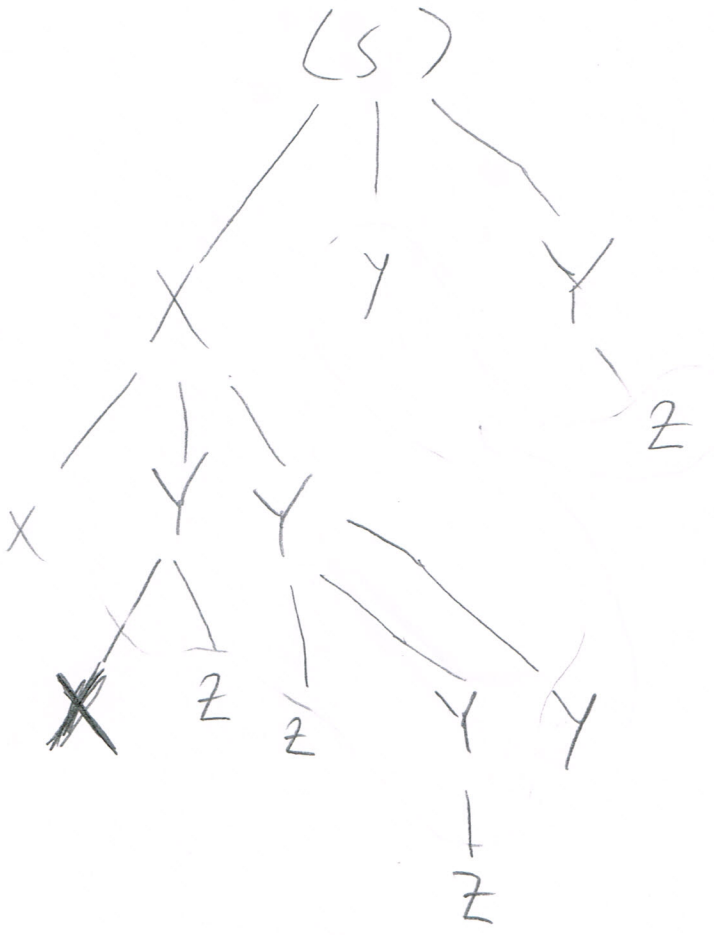
<term> -> <factor> { (* | / | %) <factor>)

<factor> -> id | int constant | (<expr>)

<while cond true> -> <if statment> <assign>

<if statment> -> if (<cond>) <assign>

6) xXzzzyy z



capital Y

lower case y

phrases: xXzzzyz, xX zzzy, Xz, zzy, z, z

simple phrases: z, Xz

Handle: Xz

7)

Given grammar

1. E -> E+T
2. E -> T
3. T -> T*F
4. T -> F
5. F -> (E)
6. F -> Id

input : id + (id * id)

Stack	Input	Action
0	id + (id * id) $	shift 5
0 id5	+ (id * id) $	reduce 6, use Goto (0,F)
0 F3	+ (id * id) $	reduce 4, use Goto (0,T)
0 T2	+ (id * id) $	reduce 2, use Goto (0,E)
0 E1	+ (id * id) $	shift 6
0 E1 +6	(id * id) $	shift 4
0 E1 +6 (4	id * id) $	shift 5
0 E1 +6 (4 id5	* id) $	reduce 6, use Goto (4,F)
0 E1 +6 (4 F3	* id) $	reduce 4, use Goto (4,T)
0 E1 +6 (4 T2	* id) $	shift 7
0 E1 +6 (4 T 2 * 7	id) $	shift 5
0 E1 +6 (4 T2 * 7 id5	) $	reduce 6, use Goto (7,F)
0 E1 +6 (4 T2 * 7 F10	) $	reduce 3, use Goto (4,T)
0 E1 +6 (4 T2	) $	reduce 2, use Goto (4,E)
0 E1 +6 (4 E8	) $	shift 11
0 E1 +6 (4 E8) 11	$	reduce 5, use Goto (6,F)
0 E1 +6 F3	$	reduce 4, use Goto (6,T)