Kaleb Burris
CS 331 - Programming Languages
Assignment 1

---

## Exercise A

"BE SURE TO DRINK YOUR OVALTINE"

## Exercise B

:)

## Exercise C

1. Although the use of `let` allows for more dynamic typing, Swift primarily uses static typing.
2. This means that type checking is done in the static context - before runtime. Errors about invalid typing are thrown before the program ever executes.

## Exercise D

The grammar can be also represented with the regular expression $ro*z+$

The grammar contains the strings 1, 3, and 5.

## Exercise E

This language contains all strings that start with $n$ many pairs of $a$'s ($aa$), where $n$ is greater than or equal to 0, followed by zero or more $c$'s, and ended by $n$ many pairs of $b$'s ($bb$).

## Exercise F

The regular expression $x*(ab \mid c)*$ contains the strings labeled 1, 3, 4, 5, and 7.

## Exercise G

This regular expression matches all strings that have any number of $a$'s, $b$'s, and $c$'s and also have one $b$, by matching the arbitrary set of $[abc]$ on either side of a singular $b$.

$$(a \mid b \mid c)*b(a \mid b \mid c)*$$

## Exercise H

1. No, this grammar contains the rule $S \to SaS$ which is not one of the accepted forms of productions for a regular grammar. We have to track how many $S$'s are generated on both sides of the $a$.
2. Yes, this grammar is context-free as all of its productions' left sides contain a single non-terminal.

3. Leftmost:

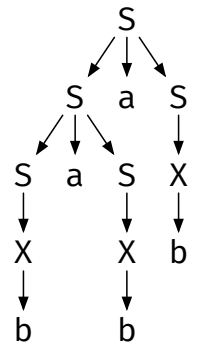
$$\boxed{\begin{aligned} &S \\ &SaS \\ &baS \\ &bab \end{aligned}}$$

4. Rightmost:

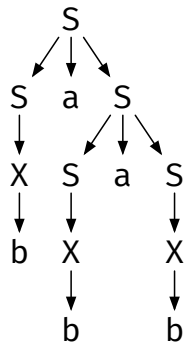
$$\boxed{\begin{aligned} &S \\ &SaS \\ &Sab \\ &bab \end{aligned}}$$

5. The string $babab$ has two different parse trees.

Parse tree one:




Parse tree two:



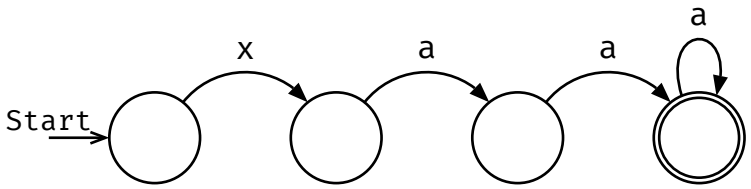

6. $$\boxed{\begin{aligned} &S \to bA \\ &A \to aS \mid \varepsilon \end{aligned}}$$

This is non-ambiguous as all derivations generate right-most derivations.

## Exercise E

1. $$\boxed{xaa+}$$

2.




3. $$\boxed{\begin{aligned} &S \to xaaA \\ &A \to a \mid \varepsilon \end{aligned}}$$

4. No, the grammar is not ambiguous as any derivation will always contain the first step, followed by $n$ many additional of the second steps for any number of $a$'s needed past two $a$'s. Since there are

only two possible productions for $A$, and one of them is the empty string $\varepsilon$, there are no alternative ways to construct a string in the language from this grammar.

## Exercise J

The provided statements are given (NOT IN BNF FORM!!!):

<uc-letter> = A-Z
<lc-letter> = a-z

My solution is:

```
        <name> = <real-name> <middle-name> <real-name>
   <real-name> = <uc-letter> <lc-letter> <lc-tail>
     <lc-tail> = <lc-letter> <lc-tail> | ""
<middle-name> = " " | " " <uc-letter> ". "
```

Note: " " is a single space character and "" is the empty string.