Avaree Warrick
Compiler Design
Assignment 2

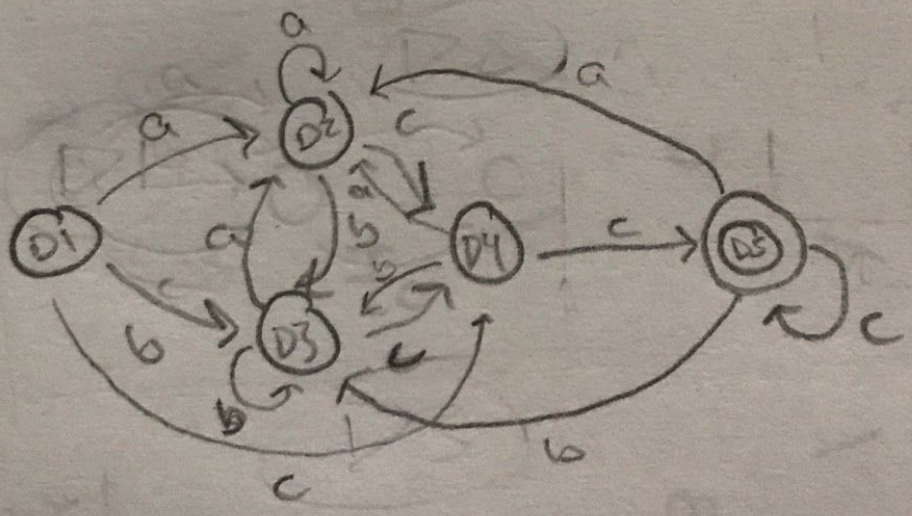
a → b → c → c

→ ending in c

1a.

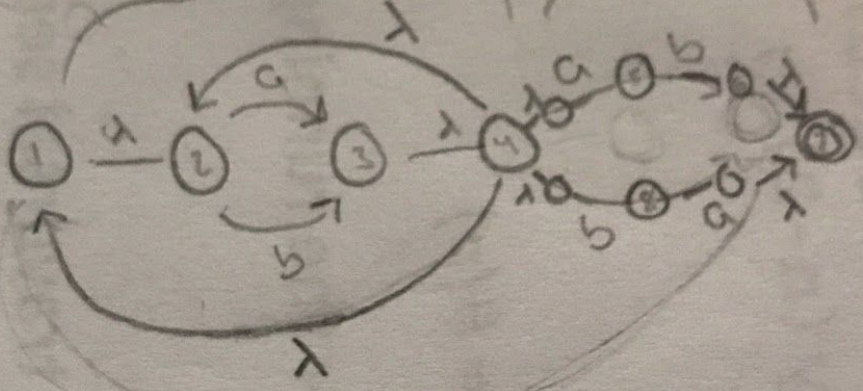
$$(a|b|c)^* \cdot (cc)$$

1b.



(a|b)* (ab|ba)

2.a (a|b)* (ab|ba)

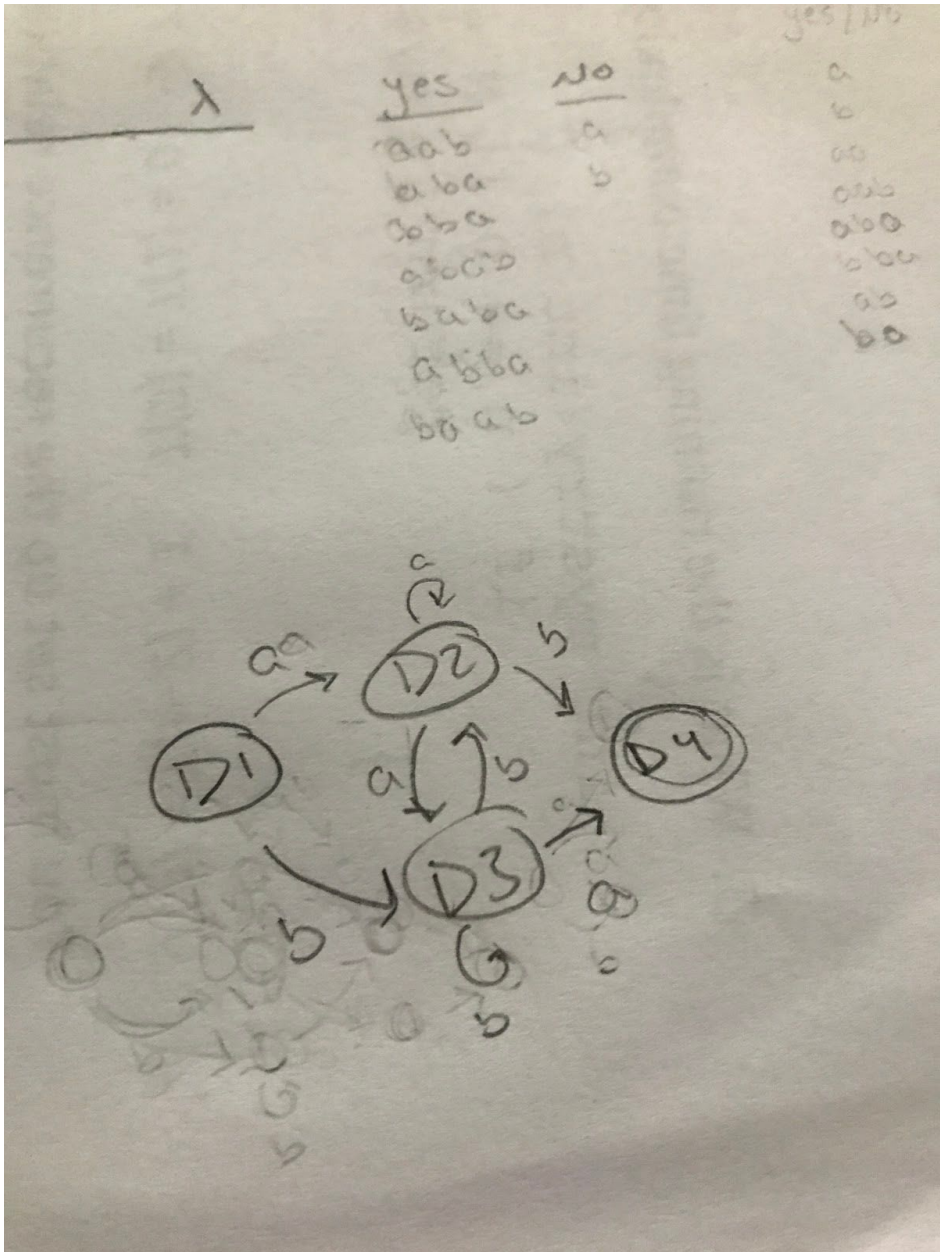


2.b

a	b	λ
3	3	

yes

aab
aba
abba
abab
baba



3.

I ended up not finishing this one but here is my code so far…

```
%{
  #include <stdlib.h>
  #include <stdio.h>
%}

%%
```

```
[ \t\n\r]                                   { }
"{"[^"}"]*"}"                               { }
[A-Za-z_][A-Za-z_0-9]*                      { printf("IDENTIFIER\t%s\n",yytext); }
[0-9]+                                      { printf("INTEGER\t\t%d\n",atoi(yytext)); }
NIL|Nil|nil                                 { printf("NIL\n"); }
"&"[^"&"]*"&"                               { printf("STRING CONSTANT %s\n",yytext); }
":="                                        { printf("ASSIGNMENT\n"); }
"="                                         { printf("EQUALS\n"); }
";"                                         { printf("SEMICOLON\n"); }
","                                         { printf("COMMA\n"); }
":"                                         { printf("COLON\n"); }
".."                                        { printf("RANGE\n"); }
"."                                         { printf("PERIOD\n"); }
"("                                         { printf("BEGIN PAREN\n"); }
")"                                         { printf("END PAREN\n"); }
"["                                         { printf("BEGIN BRACKET\n"); }
"]"                                         { printf("END BRACKET\n"); }
PROGRAM|Program|program                     { printf("PROGRAM\n"); }
CONST|Const|const                           { printf("CONST\n"); }
TYPE|Type|type                              { printf("TYPE\n"); }
VAR|Var|var                                 { printf("VAR\n"); }
PROCEDURE|Procedure|procedure               { printf("PROCEDURE\n"); }
FUNCTION|Function|function                  { printf("FUNCTION\n"); }
BEGIN|Begin|begin                           { printf("BEGIN\n"); }
END|End|end                                 { printf("END\n"); }
IF|If|if                                    { printf("IF\n"); }
THEN|Then|then                              { printf("THEN\n"); }
ELSE|Else|else                              { printf("ELSE\n"); }
NOT|Not|not                                 { printf("NOT\n"); }
AND|And|and                                 { printf("AND\n"); }
OR|Or|or                                    { printf("OR\n"); }
REPEAT|Repeat|repeat                        { printf("REPEAT\n"); }
UNTIL|Until|until                           { printf("UNTIL\n"); }
FOR|For|for                                 { printf("FOR\n"); }
TO|To|to                                    { printf("TO\n"); }
DOWNTO|Downto|downto                        { printf("DOWNTO\n"); }
WHILE|While|while                           { printf("WHILE\n"); }
DO|Do|do                                    { printf("DO\n"); }
WITH|With|with                              { printf("WITH\n"); }
CASE|Case|case                              { printf("CASE\n"); }
OF|Of|of                                    { printf("OF\n"); }
"<"                                         { printf("LESS THAN\n"); }
"<="                                        { printf("LESS THAN EQUAL TO\n"); }
">"                                         { printf("GREATER THAN\n"); }
">="                                        { printf("GREATER THAN EQUAL TO\n"); }
"<>"                                        { printf("NOT EQUAL TO\n"); }
"+"                                         { printf("PLUS\n"); }
"-"                                         { printf("MINUS\n"); }
"*"                                         { printf("MULTIPLICATION\n"); }
"/"                                         { printf("DIVISION\n"); }
DIV|Div|div                                 { printf("INTEGER DIVISION\n"); }
MOD|Mod|mod                                 { printf("INTEGER MODULUS\n"); }
IN|In|in                                    { printf("IN\n"); }
PACKED|Packed|packed                        { printf("PACKED\n"); }
ARRAY|Array|array                           { printf("ARRAY\n"); }
SET|Set|set                                 { printf("SET\n"); }
RECORD|Record|record                        { printf("RECORD\n"); }
```

```
int             yywrap     ()              { return(1); }

int             main        ()
{
  yylex();
  return(EXIT_SUCCESS);
```

It wouldn’t compile and I wasn’t sure how to fix the little errors it was giving me every time I tried to fix it.