2025 Sample Exam

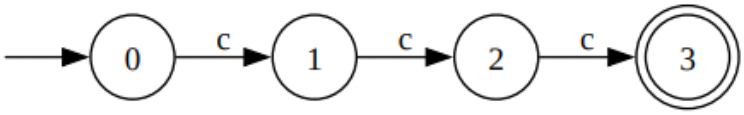
Wednesday, March 05, 2025 10:20 AM

Question 1

How many of the following 6 strings

ccc cc ccccc c cccc cccccc

are accepted, in part or whole, by the Thompson's construction nondeterministic finite state automaton shown below



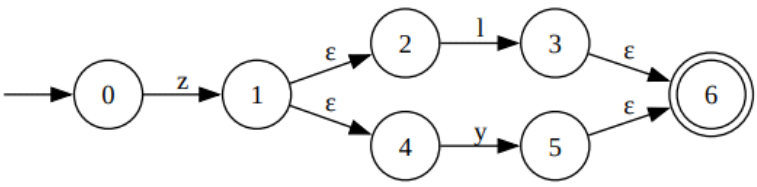
ccc -accepted
cc – not accepted
ccccc - accepted
c – not accepted
cccc - accepted
ccccc – accepted

ANS: B - 4

Question 2

llllzzzlll zzzzz yyyyyyyzz lllzzyyyy yyyyyl lllyyylll zyyyyz lllzzzyyyy yllllzzz yyylllzzzz yyyyyyyyyy yyyyyyy zzlll zlllyyy yyyyyyyy

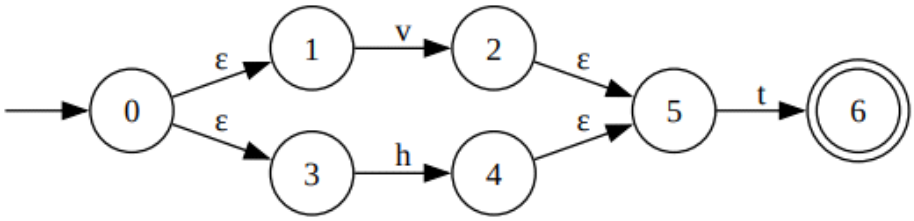
are accepted, in part or whole, by the Thompson's construction nondeterministic finite state automaton shown below



llllzzzlll - partially accepted
zzzzz - not accepted
yyyyyyyzz - not accepted
lllzzyyyy - partially accepted
yyyyyl - not accepted
lllyyylll - not accepted
zyyyyz - accepted
lllzzzyyyy - partially accepted
yllllzzz - not accepted
yyylllzzzz - not accepted
yyyyyyyyyy - not accepted
yyyyyyy - not accepted
zzlll - partially accepted
zlllyyy - accepted
yyyyyyyy - not accepted

ANS: A – 6

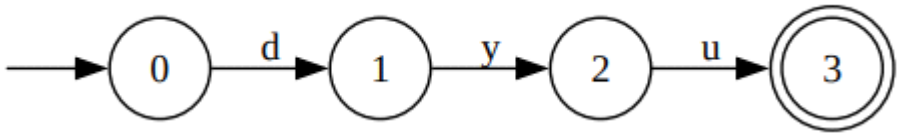
Question 3



tttttttttt - not accepted
hhhvvh - not accepted
vhhhhv - not accepted
vvvvhhh - not accepted
tttthhvvvv - not accepted
vvvvvttt - partially accepted
tvvvvt - partially accepted
vvvvhhhhvvvv - not accepted
hhhhhhh - not accepted
ttttvvvttt - partially accepted
tttthttt - partially accepted
tthv - not accepted
vvvtvvvv - partially accepted
hhhhvhhhh - not accepted
vvvttvvvv - partially accepted

ANS: E – 6

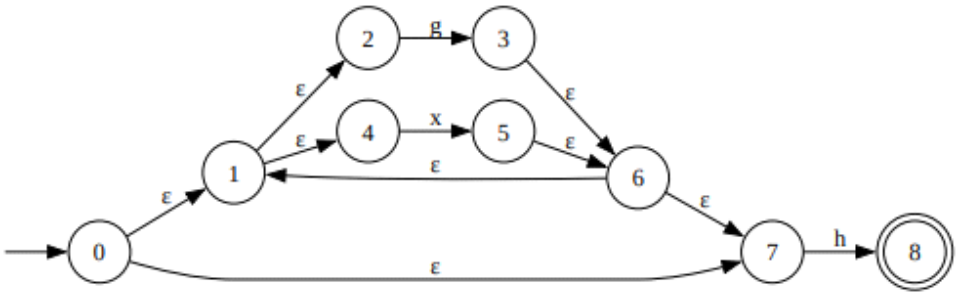
Question 4



yyyddduu - not accepted
ddduuuddd - not accepted
yyyyddddddd - not accepted
ddddddduuu - not accepted
dyyy - not accepted
ddddddddddd - not accepted
ddddyyuuuu - not accepted
uuyyyy - not accepted
uuddddy - not accepted
uuuuu - not accepted
duuuuuu - not accepted
yyyyyyyyuuuu - not accepted
uuddddduuuu - not accepted
yyyyyyuuuu - not accepted
dddduuuuu – not accepted

ANS: A – 0

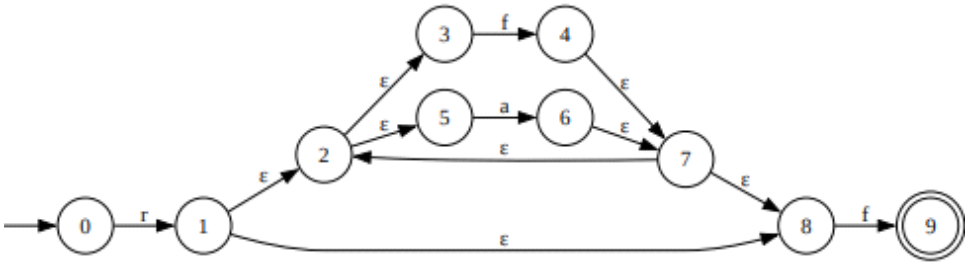
Question 5



gggxxhhhh - accepted
gxxxxhh - accepted
ggxxxxhhh - accepted
gxhh - accepted
gggxhhhh - accepted
ggggxhhhh - accepted
ggggxxxxh - accepted
gxhhh - accepted
gxxxhh - accepted
ggxxxhh - accepted
gggxxxxhhhh - accepted
ggxhhhh - accepted
ggggxxxhhh - accepted
gggxxh - accepted
gggxxxhh - accepted

ANS: D – 15

Question 6



arrrraaaa - not accepted
aaaaar - not accepted
rrrfffaaaa - accepted
aarrfff - accepted
ffaaa - not accepted
rrrrfff - accepted
fffaa - not accepted
rrraarr - not accepted
rrrrrraaa - not accepted
aaaarrrrrr - not accepted
aaarrrrrr - not accepted
ffffrrrrrrrr - not accepted
aaaaaaaa - not accepted
rrrff - accepted
aaaarrrraa – not accepted

ANS: A – 4

Question 7

```
p[a-zA-Z]J
```

The string must contain p, the next character must be any upper or lowercase letter, the next

character must be J
JJJJ – not accepted
fJJJJJ - not accepted
JJJppp - not accepted
JJffJ - not accepted
JJfffppp - not accepted
ffpJ - not accepted
fffpppf - not accepted
pJJJJ - accepted
JJJJJJppp – not accepted
ppffp - not accepted
ffJJp - not accepted
ppffff - not accepted
ppJJJf - accepted
pppf - not accepted
JJJJJp – not accepted
ANS: E - 2

Question 8

```
VVV
```

There must be 3 consecutive V's in the substring
VVVVVV - accepted
VVVV - accepted
V – not accepted
VVV - accepted
VVVVV - accepted
VV – not accepted
ANS: D- 4

Question 9

```
n[^A-Z]F
```

The string must contain the letter n, the next character must NOT be an uppercase letter, the next character must be F
tnnttt - not accepted
FFFFt - not accepted
tnnnF - accepted
FFFFFF – not accepted
nnnFF - accepted
FFFnnnF - accepted
nnttt - not accepted
nnntF - accepted
nnnnnn - not accepted
FFnnn - not accepted
nnFFF - accepted
nttttttt - not accepted
tttttFF - not accepted
nFFFF - not accepted
FnF – not accepted
ANS: C –5

Question 10

```
x[a-zA-Z][a-zA-Z]?ll
```

The string must start with x, the next character must be any upper or lowercase letter, the next character is optionally any upper or lowercase character, the next in the string must be ll

lllTx - not accepted
llllIx - not accepted
lllTxx - not accepted
lxxxT - not accepted
lTTTxxx - not accepted
llllllx - not accepted
xTTTx - not accepted
xxxllxxx - accepted
TTTTxxx - not accepted
TTTxxxx - not accepted
TTlllxx - not accepted
TTTx - not accepted
lxxlll - accepted
xxxlT - not accepted
TTTxx – not accepted
ANS: C - 2

Question 11

```
(iii|vv)
```

The string must contain iii (inclusive) or vv
vIlv - not accepted
vII - not accepted
Ivii - not accepted
viiII - not accepted
IIIIv - not accepted
vvvv - accepted
iiII - not accepted
ivii - not accepted
ivII - not accepted
IIiii - accepted
IIIv - not accepted
iivvI - accepted
iiI - not accepted
IiiI – not accepted
vvIi - accepted
ANS: D - 4

Question 12

```
d[^a-z]C
```

The string must contain d, the next character must NOT be a lowercase letter, the string must end with C
CCllllI - not accepted
CCCC – not accepted
lllllCC - not accepted
ddddd - not accepted
CCCdddd - not accepted
llllIC - not accepted
ClIC - not accepted
lCC - not accepted
CCllICC - not accepted
lddddddd - not accepted
CCdddd - not accepted
CCCCC - not accepted
ddCd - not accepted

dCCClll - accepted
llldCC - accepted
ANS: B - 2

Question 13

```
(a{2,3}|sss?)
```

The character a must appear 2 or 3 times, or optionally the string sss
QaaQ - accepted
ssa - not accepted
aaaass - accepted
QQQQss - not accepted
QQaass - accepted
Qssa - not accepted
ssQQss - not accepted
QQQQQ – not accepted
QQaaQQ - accepted
saas - accepted
aaQQ - accepted
ssaQ - not accepted
aaaaaa - accepted
sQa - not accepted
QQss - not accepted
ANS: F – OTHER
ANS: E - 13
Also applying the {2,3} quantifier to s gives the correct answer of 13, idk why???

Question 14

```
(i{2}|D{1,2}|[A-M]+)$
```

2 occurrences of i, OR 1 or 2 occurrences of D, OR one or more occurrence of any uppercase letter ranging from A-M, and there should be no extra characters after this point
kkkk - not accepted
kkii - accepted
iiDDkk - not accepted due to extra characters after the point of termination
kiik - not accepted due to extra characters after the point of termination
iikkDD - accepted
Dii - accepted
ikkkk - not accepted
DDik - not accepted due to extra characters after the point of termination
kkkkD - accepted
kii - accepted
kDii - accepted
Dkki - not accepted due to extra characters after the point of termination
iiiD - accepted
kkiikk - not accepted due to extra characters after the point of termination
kkiii - accepted
ANS: C- 8

Question 15

```
[a-z][A-Z]$
```

The string must contain any lowercase letter, followed by any uppercase letter and no more characters after this point
yyJJ - not accepted due to extra characters after the point of termination

yddyy - not accepted
yJJyy - not accepted due to extra characters after the point of termination
dddJJ - not accepted
dyydd - not accepted
dyd - not accepted
yyyd - not accepted
yyyyJJ - not accepted due to extra characters after the point of termination
yyyydd – not accepted
JJddJJ - not accepted due to extra characters after the point of termination
ddddd - not accepted
Jyyyy - not accepted
dJJyy - not accepted due to extra characters after the point of termination
dyyyy - not accepted
ydd - not accepted
ANS: C- 0

Question 16

```
..vQ.
```

The first two characters can be anything except new line breaks, there must be a v, there must be a Q, the last character can be anything except a new line break
Qvvvvv - not accepted
QQQssss - not accepted
sssvvvvv - not accepted
QQQvvQQ - accepted
QQQQvvv - not accepted
QssQQQ - not accepted
vvvvvvss - not accepted
QQQvss - not accepted
QQQQQv - not accepted
vssss - not accepted
vvsv - not accepted
QvvvQQ - accepted
vvvvvvv - not accepted
vvvssv - not accepted
QQQssQQ – not accepted
ANS: C - 2

Question 17

```
%token E
%%
sentence: E | E sentence
;
```

E is the only terminal symbol. The sentence structure is a single occurrence of E or E followed by another calling to the sentence structure

EEEE - accepted
EEE - accepted
E - accepted
EEEEEE - accepted
sentence – not accepted, not a terminal symbol
EE - accepted
MRpF5Hu – not accepted, not defined anywhere
ANS: E -5

Question 18

```
%token I
%%
sentence: I | sentence  I
;
```

BKHDqiC - not accepted, not defined anywhere
IIIII - accepted
IIII - accepted
I - accepted
II - accepted
sentence – not accepted, not a terminal symbol
ANS: B - 4

Question 19

```
%token d O
%%
sentence: sub | sub sentence
sub: d | O
;
```

sentence – not accepted, not a terminal symbol
dddOOO - accepted
sSMXMh1 – not accepted, not defined anywhere
ddddddOOOOO - accepted
ddddddOOO - accepted
ddddddO - accepted
ddOOO - accepted
ANS: B –5

Question 20

```
%token s H
%%
sentence: s | H | s sentence
;
```

sssH - accepted
ssHHHH - not accepted
sHHH - not accepted
sHHHH – not accepted
sssHHH - not accepted
ssHHH - not accepted
ssssHHHH – not accepted
sss - accepted
sssHHHH – not accepted
HH - not accepted
ANS: F – Other
ANS: A - 2 , there can only ever be one occurrence of H for it to be valid

Question 21

```
%token z G
%%
sentence: z | G | sentence z
;
```

zGG - not accepted
zzzGG - not accepted
zG - not accepted

GG - not accepted
zGGG - accepted
zzzGGGG - not accepted
zzGGG - not accepted
zzzzGGGG - not accepted
zz - accepted
zzGG – not accepted
ANS: C - 1

Question 22

```
%token x A
%%
sentence: x | A | A sentence
;
```

xxxA - not accepted
xxAAAA - not accepted
xxxxAA - not accepted
xxxx - not accepted
A - accepted
xxxAA - not accepted
xAAAA - not accepted
xxxxAAAA - not accepted
xxAAA - not accepted
xxAA – not accepted
ANS: B – 1

Question 23

```
%token s B
%%
sentence: s | B | sentence B
;
```

ssssBBB - not accepted
sBBBB - accepted
sssBBB - not accepted
ssss - not accepted
ssB - not accepted
ssBB - not accepted
sB - accepted
sBB - accepted
BB - accepted
sssBB – not accepted
ANS: D - 9
ANS: C – 4, there can only ever be one occurrence of s for the string to be valid

Question 24

```
%token I
%%
sentence: list | sentence list
list: listc ';'
listc: I | I listc
;
```

II;IIII;III – not accepted
IIII;III;II - not accepted
IIIIIII;II;II - not accepted

llllll;llllll - not accepted
l;llllll - not accepted
lllllllll - not accepted
ANS: C - 0

Question 25

```
%token M
%%
sentence: listc | listc ',' sentence
listc: M | M listc
;
```

MMMMMMM,MMMMM, - not accepted
MM,MMMMMMMMM, - not accepted
MMM,M,MMM - accepted
M,MMMMMMMM,MM - accepted
MMM,M,MM,MMM, - not accepted
ANS: A - 2

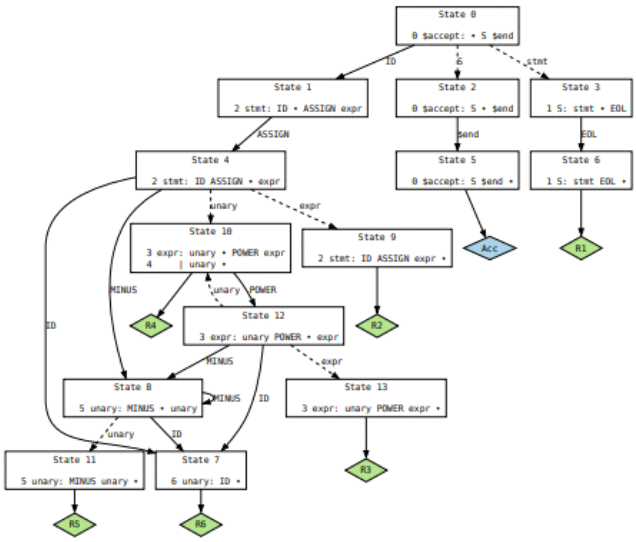
Question 26

```
%token l
%%
sentence: commal ';'
commal: listc | listc ',' commal
listc: l | l listc
;
```

lll,l,llll,ll - not accepted
llll,lll,llllllll,llll, - not accepted
lllllllll; - accepted
lllllllll,lllllll - not accepted
llll,llllll,l; - accepted
ll,ll - not accepted
lllllllll; - accepted
ANS: E – 3

Question 27

```
Given the following tokens
"^"     { return POWER; }
"-"     { return MINUS; }
":="    { return ASSIGN; }
[a-z]   { yylval = yytext[0]; return ID; }
\n      { return EOL; }
and the following Bison Context Free Grammar
    0 $accept: S $end
    1 S: stmt EOL
    2 stmt: ID ASSIGN expr
    3 expr: unary POWER expr
    4     | unary
    5 unary: MINUS unary
    6      | ID
which generates the Bison Shift Reduce Parser
```



What sequence of states will the Bison Shift Reduce Parser go through parsing the sentence a:=b^c \n

0: input token "a", "a" is an ID, on ID, shift to State 1.
The stack has [0,1 - ID]

1: input token ":=", ":=" is an ASSIGN, shift to state 4.
The stack has [0,1 - ID, 4 – ASSIGN]

4: input token "b", "b" is an ID, shift to state 7.
The stack has [0,1 - ID, 4 – ASSIGN ,7 – ID]

7: input token "^", state 7 reduces by rule six (ID -> unary), By popping 7 from the stack, we are back in state 4 and can move to state 10 with our unary input.
The stack has [0,1 - ID, 4 – ASSIGN ,10 – UNARY]

10: input token "^", "^" is a POWER, shift to state 12.
The stack has [0,1 - ID, 4 – ASSIGN ,10 – UNARY ,12- POWER]

12: input token "c", "c" is an ID, shift to state 7.
The stack has [0,1 - ID, 4 – ASSIGN , 10 – UNARY , 12- POWER, 7 – ID]

7: input token "\n", state 7 reduces by rule six (ID -> unary), By popping 7 from the stack we are back in state 12. We must follow the dotted lines to state 10 to act on a unary expression.
The stack is [0,1 – ID ,4 – ASSIGN , 10 – UNARY , 12- POWER, 10 - UNARY]

10: input token "\n", we cannot consume the EOL token in this state.
We must perform another reduction or else the string is invalid.
Rule three in this state says that "expr" can be made up of unary, power, expr.
Rule four in this state says that "expr" can be made up of unary.

The top of the stack currently reads [10 unary, 12 power, 10 unary].
Rule three cannot apply here because we do not have unary, power, expr.

Follow rule four to pop [10] from the stack.
The stack is now [0, 1 – ID ,4 – ASSIGN , 10 – UNARY , 12- POWER].
From state 12, we can get to state 13, using the expr we have just generated from rule four.
The stack has [0, 1 – ID ,4 – ASSIGN , 10 – UNARY , 12- POWER, 13- EXPR].

13: input token "\n", we cannot consume EOL token in this state.
We must reduce in this state according to rule three.
Rule three says that "expr" can be made up of unary, power, expr.
The top of the stack currently reads [10 unary, 12 power, 13 expr].
We can pop these off the stack.
The stack without them reads [0, 1 – ID , 4 – ASSIGN]
Creating an expr in state 4 leads us to state 9.
The stack has [0, 1 – ID , 4 – ASSIGN, 9- EXPR]

9: input token "\n", we cannot consume EOL token in this state.
We must reduce in this state according to rule two.
Rule two says that a stmt can be created from ID ASSIGN EXPR.
The top of the stack currently reads [1 – ID , 4 – ASSIGN, 9- EXPR]
We can pop these off the stack.
The stack now has [0].
From state 0 and the stmt we have just generated, we can move to state 3.
the stack has [0, 3 – stmt].

3: input token "\n", "\n" is an EOL, shift to state 6.
The stack has [0, 3- stmt, 6 – EOL].

6: State 6 says to reduce according to rule one.
Rule one says that an "S" can be created from a stmt and EOL.
The top of the stack currently has [3- stmt, 6 – EOL].
We can pop this off the stack.
From state 0 and the S we have just created, we can shift to state 2.
The stack has [0, 2 – S].

2: State 2 says to reduce according to rule zero.
Rule zero says that an $end token can be generated with an S.
The top of the stack has an S.

5: State 5 is the accepting state. The string has been parsed successfully.

*not included in the answer

2023

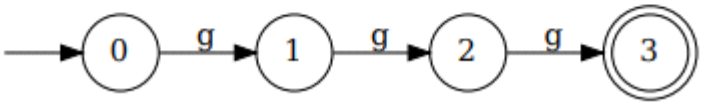
Friday, January 31, 2025 10:13 AM

Q 1.

How many of the following 6 strings

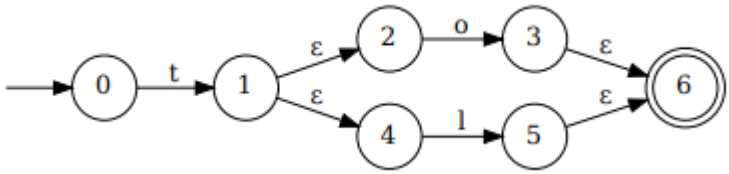
g gg gggg ggggg gggggg ggg

are accepted, in part or whole, by the Thompson's construction nondeterministic finite state automaton shown below

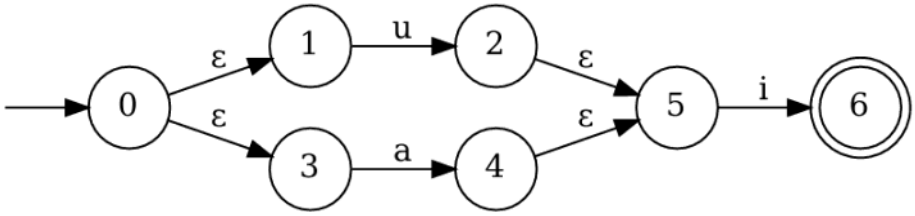


g : not accepted, ends in state 1
gg : not accepted, ends in state 2
gggg : accepted, ends in state 3, there are no more transitions to be made but 3 is an accepting state
ggggg : accepted, ends in state 3, there are no more transitions to be made but 3 is an accepting state
gggggg : accepted, ends in state 3, there are no more transitions to be made but 3 is an accepting state
ggg : accepted, ends in state 3

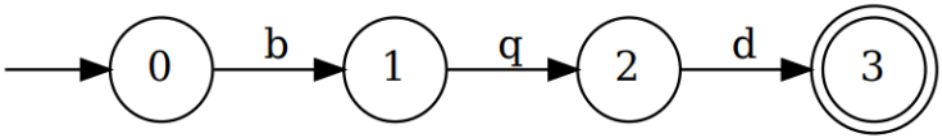
Ans: 4 - C



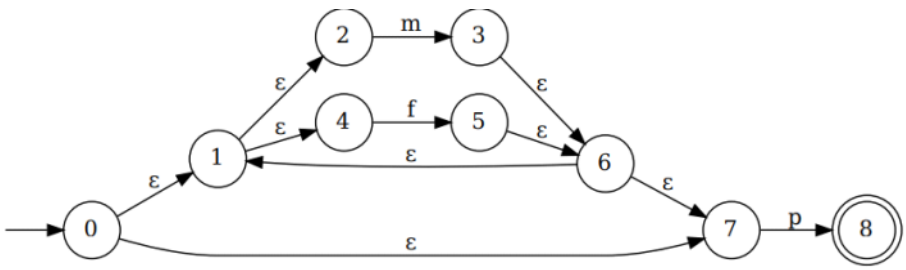
ttttto - accepted
llllottt
lllttttll - accepted
llltlll - accepted
ttllltt - accepted
oloo
loo
ttottt - accepted
lllltttll - accepted
ootll - accepted
ooolo
oooollllttt
tttoooolll - accepted
llllll
ttttllllltt - accepted
Ans: 9 – B



uiiiaa - accepted
iiiaii - accepted
uuuuuuu
uuuuiiii - accepted
iiiiiiiii
aaauuu
iiiuuuuiiii - accepted
iiiiaaaaii - accepted
iiiaaaa
uuuuaaaii - accepted
aaauuuuuuuu
uuaaaaiii - accepted
iuuuuaaa
uau
aaaaaii- accepted
Ans: 8 - D

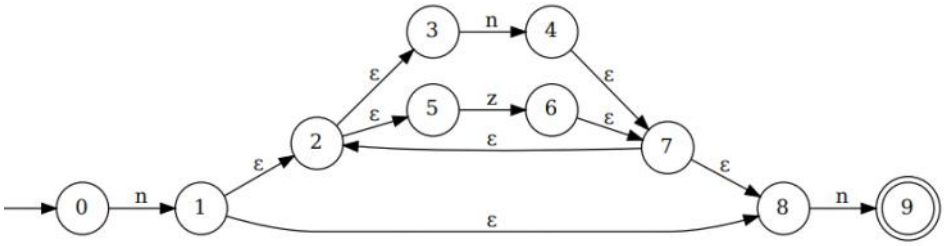


ddddbbbb
qqbbbbbb
qqqqbbb
qqqqdqq
dbbb
qdddd
dddddqqqq
bbbdbb
bbbbbbbbbb
dddqdddd
qbbddd
qqqqbb
ddbbbb
qqqddddqqq
dqqqb
Ans: 0 - A



mmffpppp - accepted
mmfpppp - accepted
mmmmfffpppp - accepted
mmmffpppp - accepted

mfp - accepted
mmmmffffpp -accepted
mmmmfffp - accepted
mffffpp - accepted
mmmmfpp - accepted
mmfffp - accepted
mmffpp - accepted
mmmffffppp - accepted
mmffp - accepted
mmmmffffppp - accepted
mffffppp- accepted
Ans: 15 - A



nnnnnnnn - accepted
nnznn - accepted
nnnnnnnzz - accepted
nnnzzzznnn - accepted
zznnzzz
nnnnnnn - accepted
nnnnnnnnnn - accepted
nnnn - accepted
nnnnnn - accepted
zznn
nnnzzznnnn - accepted
nnnnnnnz - accepted
zzznnnnn - accepted
zzznn
nnnzz - accepted
Ans: 12 - D

How many of the following 6 strings jjj jjjj jj jjjjjj jjjjj j are matched at least once, in part or whole, by the Flex regular expression jjj
jjj - accepted
jjjj - accepted
jj
jjjjjj - accepted
jjjjj - accepted
j
Ans: 4 - A

How many of the following 15 strings

JJJbbbbb tttbbJJ bbbJJttt ttJbb bbJJbbb JJttJ JJbbbJJ bbtt JJttt bbJJJt tJJJJ bbbbbttt tttttbb bbbJJJJJ JJtttt

are matched at least once, in part or whole, by the Flex regular expression

t[^a-z]J

JJJbbbbb

tttbbJJ

bbbJJttt

ttJbb

bbJJbbb

JJttJ

JJbbbJJ

bbtt

JJttt

bbJJJt

tJJJJ - accepted

bbbbbttt

tttttbb

bbbJJJJJ

JJtttt

Ans: 1 - B

How many of the following 15 strings

ccccc cVVVc uuucc ccVVuu ccccuuu cccccVVV VVVuuuuu uuuV uuuuuu VcVV VVuuu VVVV uuuVVV cVVVu VVVVuu are matched at least once, in part or whole, by the Flex regular expression

c[^A-Z]V

ccccc

cVVVc

uuucc

ccVVuu - accepted

ccccuuu

ccccVV - accepted

VVVuuuuu

uuuV

uuuuuu

VcVV

VVuuu

VVVV

uuuVVV

cVVVu

VVVVuu

Ans: 2 - B

How many of the following 15 strings kkkKKyyy KKkkyy kkkKK yyyykkk KKKKk yKkkk KyyyKKK yykkK kkKKKkkk kkKy kyk kkkkKK kkkkyy yykk KKKyyykk are matched at least once, in part or whole, by the Flex regular expression
yy[a-zA-Z][a-zA-Z]+k

kkkKKyyy

KKkkyy

kkkKK

yyyykkk - accepted

KKKKk

yKkkk

KyyyKKK

yykkK

kkKKKkkk

kkKy

kyk

kkkkKK

kkkkyy

yykk

KKKyyykk- accepted

Ans: 2 - D

How many of the following 15 strings

nnnnnN hhhhnnn hNNN NNnnnnn hhhhhNN nnnNNNN nNNh nnnNN NnnNNN hhhnh hhhnnn nnnnNN nnnhhhN NNNnh hhhhhhh are matched at least once, in part or whole, by the Flex regular expression

h[a-zA-Z][a-zA-Z]?nn

nnnnnN

hhhhnnn

hNNN

NNnnnnn

hhhhhNN

nnnNNNN

nNNh

nnnNN

NnnNNN

hhhnh

hhhnnn

nnnnNN

nnnhhhN

NNNnh

hhhhhhh

Ans: 7 - OTHER

How many of the following 15 strings hhhhh RRRR ffRRff fffhhh hhffhh fffff RRRhhhRR hhffffff RRRRR RRfh fffffffff fffffhhh hRRRhh Rffff hRRf are matched at least once, in part or whole, by the Flex regular expression

h[a-zA-Z]{1,2}ff

hhhhh

RRRR

ffRRff

fffhhh

hhffhh

fffff

RRRhhhRR

hhffffff

RRRRR

RRfh

fffffffff

ffffffhhh

hRRRhh

Rffff

HRRf

Ans: 2 - B

How many of the following 15 strings VVVnn VVV vvVnn vvVVnn Vvnn nnVVV vVVVV vvvvV VVVVv vvnn VnnVV vnnv VVvVV vnV VVVv are matched at least once, in part or whole, by the Flex regular expression

([A-Z]{2,3}|[a-z]{4})

VVVnn

VVV

vvVnn

vvVVnn

Vvnn

nnVVV

vVVVV

vvvvV

VVVVv

vvnn

VnnVV

vnnv

VVvVV

vnV

VVVv

Ans: 12 - B

How many of the following 15 strings LLeeee eeeeb LbL LLLeeLL bbbbbee bbbbb eeeLLLeee eeeLee Lbbeee LLeebb eebbbb bbLe LLLLeee LLLLb eeeLLL are matched at least once, in part or whole, by the Flex regular expression

ee.

LLeeee

eeeeb

LbL

LLLeeLL

bbbbbee

bbbbb

eeeLLLeee

eeeLee

Lbbeee

LLeebb

eebbbb

bbLe

LLLLeee

LLLLb

eeeLLL

Ans: 10 - E

How many of the following 15 strings qkk kJJ qJq kJk kkq JqJ Jqk kqq qkq qJJ JJk kkk qqk Jqq Jkk are matched at least once, in part or whole, by the Flex regular expression
^[a-z]

qkk

kJJ

qJq

kJk

kkq

JqJ

Jqk

kqq

qkq

qJJ

JJk

kkk

qqk

Jqq

Jkk

Ans: 15 - A

How many of the following 15 strings are matched at least once, in part or whole, by the Flex regular expression
^(w|R|[A-Z]+)

mRw
mmR
wRRw
wwwww
wwRRm
RRRRmm
mmRRR

RRmmww
RRwm
RRRR
mRmm
mmmm
Rmm
wRRmm
RwwR
Ans: 14 - C

How many of the following 7 sentences AA AAAAA AAAA AAAAAA sentence A 2R9kLOA are in the language defined by the Bison Context Free Grammar
%token A
%%
sentence: A | A sentence
;
AA - accepted
AAAAA - accepted
AAAA - accepted
AAAAAA - accepted
sentence
A - accepted
2R9kLOA
Ans: 5 - D

How many of the following 8 sentences n VtwC8bL nnnnn nnnn nnn nn sentence nnnnnnn are in the language defined by the Bison Context Free Grammar
%token n
%%
sentence: n | sentence n
;
n - accepted
VtwC8bL
nnnnn - accepted
nnnn - accepted
nnn - accepted
nn - accepted
sentence
nnnnnnn - accepted
Ans: 6 - D

How many of the following 7 sentences qFFFFF qqqFFFFF qqqFFFF jR6D9XK qqqqqqFFFFFFF sentence qqqqFFF are in the language defined by the Bison Context Free Grammar
%token q F
%%
sentence: sub | sub sentence
sub: q | F
;
qFFFFF - accepted
qqqFFFFF - accepted
qqqFFFF - accepted
jR6D9XK
qqqqqqFFFFFFF - accepted
sentence
qqqqFFF - accepted
Ans: 5 - D

How many of the following 10 sentences rrrX rXX rrX XX rrrrXXXX rrrXX rrrXXXX rXXX rr rrrrXX are in the language defined by the Bison Context Free Grammar
%token r X
%%
sentence: r | X | r sentence
;
rrrX - accepted
rXX
rrX - accepted
XX
rrrrXXXX
rrrXX
rrrXXXX
rXXX
rr - accepted
rrrrXX
Ans: 3 - E

How many of the following 10 sentences mUUUU mUUU mmmmUUU mmmmUUUU mmmU mmmUU mmU mm UUU mmUU are in the language defined by the Bison Context Free Grammar
%token m U
%%
sentence: m | U | sentence m
;
mUUUU
mUUU
mmmmUUU
mmmmUUUU
mmmU
mmmUU
mmU
mm - accepted
UUU
mmUU
Ans: 1 - A

How many of the following 10 sentences uC uuCCCC uuuC uuuuCCC uuuuCC uuuuCCCC CCC uuuu uCCCC uuuCCC are in the language defined by the Bison Context Free Grammar
%token u C
%%
sentence: u | C | C sentence
;
uC
uuCCCC
uuuC
uuuuCCC
uuuuCC
uuuuCCCC
CCC - accepted
uuuu
uCCCC
uuuCCC
Ans: 1 - B

How many of the following 10 sentences mmmmLL mmmmL mmmmLLL mLLLL L mmmm mmLLL mmmLLL mmmmLLLL mLLL are in the language defined by the Bison Context Free Grammar
%token m L

%%
sentence: m | L | sentence L ;
mmmmLL - accepted
mmmmL - accepted
mmmmLLL - accepted
mLLLL - accepted
L - accepted
mmmm
mmLLL - accepted
mmmLLL - accepted
mmmmLLLL - accepted
mLLL - accepted
Ans: 9 - A

How many of the following 6 sentences UUUU;UUUU;UUUU;UUU;UUUU; UUUUUUUU; UU;UUUUUUUUU; UU;UUUUUUUUUU; UUU;U;UU; UUUUUUU;UUUUU;UUUU; are in the language defined by the Bison Context Free Grammar
%token U
%%
sentence: list | sentence list
list: listc ';'
listc: U | U listc
;
UUUU;UUUU;UUUU;UUU;UUUU; - accepted
UUUUUUUU; - accepted
UU;UUUUUUUUU; - accepted
UU;UUUUUUUUUU; - accepted
UUU;U;UU; - accepted
UUUUUUU;UUUUU;UUUU; - accepted
Ans: 6 - D

How many of the following 5 sentences iii,iii,i, ii,iiiii,iiiii,iiiii i,iiiii, iiiiiii,iii, iiiiiiiiii,i, are in the language defined by the Bison Context Free Grammar
%token i
%%
sentence: listc | listc ',' sentence
listc: i | i listc
;
iii,iii,i,
ii,iiiii,iiiii,iiiii - accepted
i,iiiii,
iiiiiii,iii,
iiiiiiiiii,i,
Ans: 1 - C

How many of the following 7 sentences NNNNN,; N,NN NNN,; NNNN; NN,N,N NN,N,N,N, N,NN, are in the language defined by the Bison Context Free Grammar
%token N
%%
sentence: commal ';'
commal: listc | listc ',' commal
listc: N | N listc
;
NNNNN,;
N,NN
NNN,; x
NNNN; - accepted

NN,N,N
NN,N,N,N,
N,NN,
Ans: 1 – D

Q 27.

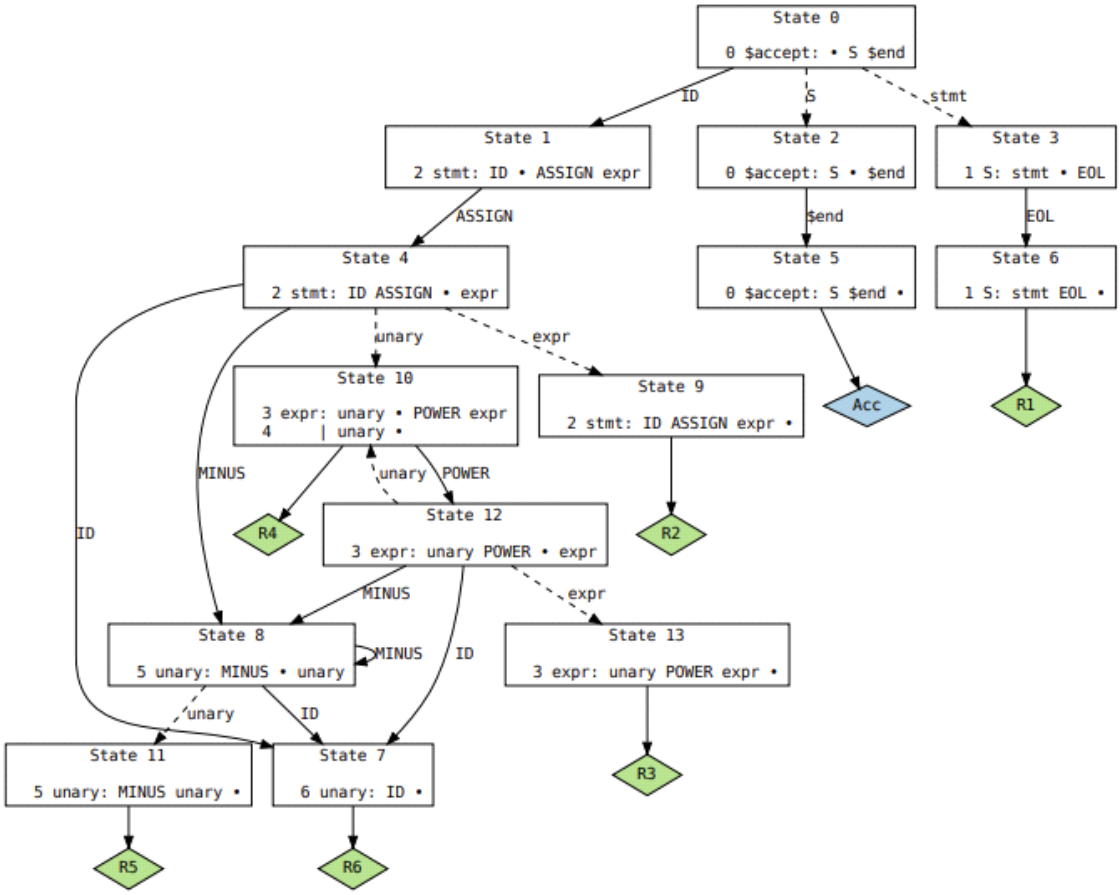
Given the following tokens

```
"^"     { return POWER; }
"-"     { return MINUS; }
":="    { return ASSIGN; }
[a-z]   { yylval = yytext[0]; return ID; }
\n      { return EOL; }
```

and the following Bison Context Free Grammar

```
    0 $accept: S $end
    1 S: stmt EOL
    2 stmt: ID ASSIGN expr
    3 expr: unary POWER expr
    4     | unary
    5 unary: MINUS unary
    6      | ID
```

which generates the Bison Shift Reduce Parser



What sequence of states will the Bison Shift Reduce Parser go through parsing the sentence
a:=b^c^d\n

0: input token "a". "a" is an ID. Move to state 1.
The stack has [0, 1-ID]

1: input token ":=". ":=" is an ASSIGN. Shift to state 4.
The stack has [0, 1-ID, 4-ASSIGN].

4: input token "b". "b" is an ID. Shift to state 7.
The stack has [0, 1-ID, 4-ASSIGN, 7-ID].

7: input token "^". Cannot be consumed in this state.
Reduce according to rule six.
A unary can be generated from an ID.
The top of the stack has [7-ID].
Pop this from the stack.
Then from state 4 and the unary we have just created, we can shift to state 10.

10: input token "^". Shift to state 12.
The stack has [0, 1-ID, 4-ASSIGN, 10- unary, 12-POWER]

12: input token "c". "c" is an ID. Shift to state 7.
The stack has [0, 1-ID, 4-ASSIGN, 10- unary, 12-POWER, 7-ID]

7: input token "^". Cannot be consumed in this state.
Reduce according to rule six.
A unary can be generated from an ID.
The top of the stack has [7-ID].
From state 12 and the unary we have just generated, we shift back to state 10.
The stack has [0, 1-ID, 4-ASSIGN, 10- unary, 12-POWER, 10-unary].

10: input token "^". Shift to state 12.
The stack has [0, 1-ID, 4-ASSIGN, 10- unary, 12-POWER, 10-unary, 12-POWER].

12: input token "d". "d" is an ID. Shift to state 7.
The stack has [0, 1-ID, 4-ASSIGN, 10- unary, 12-POWER, 10-unary, 12-POWER, 7-ID].

7: input token "\n". Cannot be consumed in this state.
Reduce according to rule six.
A unary can be generated from an ID.
The top of the stack has [7-ID].
From state 12 and the unary we have just generated, we shift back to state 10.
The stack has [0, 1-ID, 4-ASSIGN, 10- unary, 12-POWER, 10-unary, 12-POWER, 10-unary].

10: input token "\n". Cannot be consumed in this state.
Reduce according to rule four by popping 10-unary off the stack.
The stack has [0, 1-ID, 4-ASSIGN, 10- unary, 12-POWER, 10-unary, 12-POWER].
From state 12 and the expr we have just created, we can move to state 13.
The stack has [0, 1-ID, 4-ASSIGN, 10- unary, 12-POWER, 10-unary, 12-POWER, 13-expr].

13: input token "\n". Cannot be consumed in this state.
Reduce according to rule three by popping [10-unary, 12-POWER, 13-expr] off the stack.
The stack has [0, 1-ID, 4-ASSIGN, 10- unary, 12-POWER].
From state 12 and the expr we have just created. We shift to state 13.
The stack has [0, 1-ID, 4-ASSIGN, 10- unary, 12-POWER, 13-expr]

13: input token "\n". Cannot be consumed in this state.
Reduce according to rule three by popping [10-unary, 12-POWER, 13-expr] off the stack.

The stack has [0, 1-ID, 4-ASSIGN].
From state 4 and the expr we have just created. We shift to state 9.
The stack has [0, 1-ID, 4-ASSIGN, 9-expr]

9: input token "\n". Cannot be consumed in this state.
Reduce according to rule two by popping [1-ID, 4-ASSIGN, 9-expr] off the stack.
From state 0 and the stmt we have just created. Shift to state 3.
The stack has [0, 3-stmt].

3: input token "\n". Shift to state 6.
The stack has [0, 3-stmt, 6-EOL].

6: Reduce according to rule one by popping [3-stmt, 6-EOL] off the stack.
From state 0 and the S we have just created, shift to state 2.

2: Reduce according to rule zero. Shift to state 5.

5: The accepting state.

2023 Sample Exam

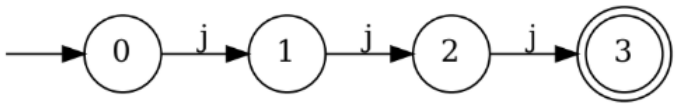
Tuesday, April 22, 2025 11:28 AM

Q 1.

How many of the following 6 strings

jjj j jjjjj jj jjjj jjjjjj

are accepted, in part or whole, by the Thompson's construction nondeterministic finite state automaton shown below



(A) 2 (B) 6 (C) 4 (D) 1 (E) 5 (F) OTHER

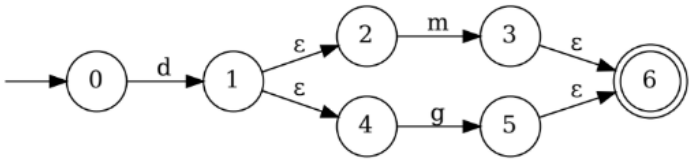
jjj: accepted
j:
jjjjj: accepted
jj:
jjjj: accepted
jjjjjj: accepted
ANS: C-4

Q 2.

How many of the following 15 strings

ggggggmm mmmgg mmmggggg mgggmmmm ddddddddddd mmmmmmm dmmgggg dddddg ddddgmm dddmmmgg ggggdddgg ddddmmmm gggggggg ddddddgg dddd

are accepted, in part or whole, by the Thompson's construction nondeterministic finite state automaton shown below



(A) 13 (B) 12 (C) 7 (D) 4 (E) 14 (F) OTHER

ggggggmm
mmmgg
mmmggggg
mgggmmmm
ddddddddddd
mmmmmmm
dmmgggg
ddddg
dddgmm
ddmmmgg

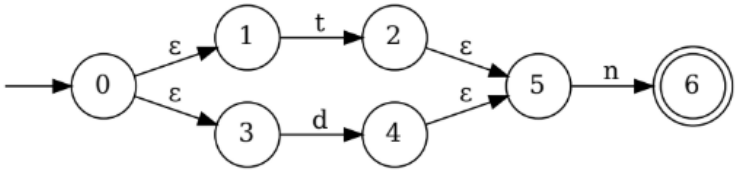
ggggddgg
dddmmmm
gggggggg
dddddgg
Dddd
ANS: C-7

Q 3.

How many of the following 15 strings

ttddddtttt ddnnnn nttttnn dddddtt ttnnd ndddd tdddt tnnnnddddd
ttttddttt dddddnnnn ttnt nntnnnn ttdd nnnnddddttt ddtttt

are accepted, in part or whole, by the Thompson's construction nondeterministic finite state automaton shown below



(A) 3 (B) 10 (C) 6 (D) 12 (E) 7 (F) OTHER

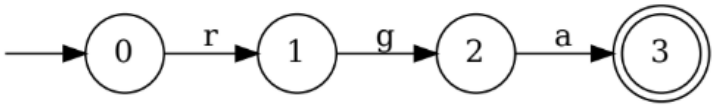
ttddddtttt
ddnnnn
nttttnn
dddddtt
ttnnd
ndddd
tdddt
tnnnnddddd
ttttddttt
ddddnnnn
ttnt
nntnnnn
ttdd
nnnnddddttt
Ddtttt
ANS: E-7

Q 4.

How many of the following 15 strings

aaaaaaa aaaargg grrrrrrr ggrrgg ggggggrrr rrrraaaggg aagaa
rrrrgggggg grrrrgg agaa raggg ggggagggg rrrrrrgggg rrraaa aaaaar

are accepted, in part or whole, by the Thompson's construction
nondeterministic finite state automaton shown below



(A) 14 (B) 0 (C) 11 (D) 9 (E) 3 (F) OTHER

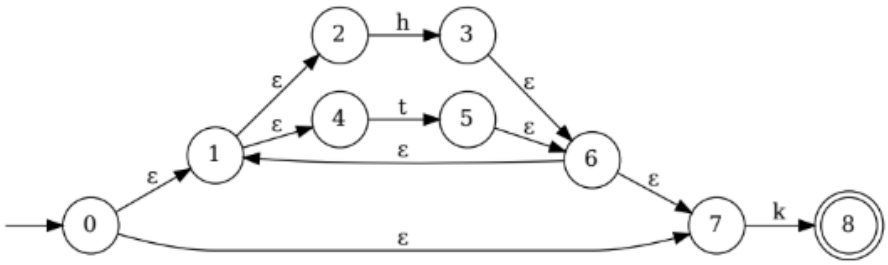
aaaaaaa
aaaargg
grrrrrrr
ggrrgg
ggggggrrr
rrrraaaggg
aagaa
rrrrgggggg
grrrrgg
agaa
raggg
ggggagggg
rrrrrrgggg
rrraaa
Aaaaar
ANS: B-0

Q 5.

How many of the following 15 strings

hhtttk hhtkkkk hhtttkkk hhhttk hhhhtk hhhtkkk httkkk hhhtttkk
hhhhtttttkk hhttttk hhhtkkkk hhhhttkk hhhttkk hhhhtttkkkk httkkkk

are accepted, in part or whole, by the Thompson's construction
nondeterministic finite state automaton shown below



(A) 2 (B) 15 (C) 10 (D) 4 (E) 6 (F) OTHER

hhtttk
hhtkkkk

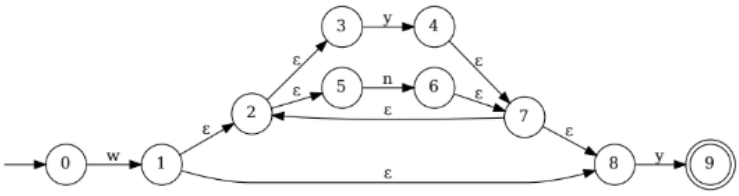
hhtttkkk
hhhttk
hhhhtk
hhhtkkk
httkkk
hhhtttkk
hhhhttttkk
hhttttk
hhhtkkkk
hhhhttkk
hhhttkk
hhhhttttkkkk
httkkkk
ANS: B-15

Q 6.

How many of the following 15 strings

yyyywww yyyyyywwww nwwwnnnn www nnnnnyyyy wwwwwnnnn nnnnnnyyyy
wwwwyyyyyyy yyyynnw yyyywwwwnnnn nwwwww ywwwwww nnnnnyyy nnnwwwnnnn
wwwwwnn

are accepted, in part or whole, by the Thompson's construction
nondeterministic finite state automaton shown below



(A) 3 (B) 11 (C) 14 (D) 1 (E) 12 (F) OTHER

yyyywww
yyyyyywwww
nwwwnnnn
www
nnnnnyyyy
wwwwwnnnn
nnnnnnnyyyy

wwwwyyyyyyy

yyyynnw

yyyywwwwnnnn

nwwwww

ywwwwww

nnnnnyyy

nnnwwwnnnn

Wwwwwnn
ANS: D-1

Q 7.

How many of the following 6 strings

P PP PPPPPP PPP PPPPP PPPP

are matched at least once, in part or whole, by the Flex regular expression

PPP

(A) 4 (B) 3 (C) 1 (D) 5 (E) 2 (F) OTHER

P
PP
PPPPPP
PPP
PPPPP
PPPP
ANS: A-4

Q 8.

How many of the following 15 strings

nBBBn nnnnnnn ccccccc BBBBBB ncccc nBB cccccc BBBnnBB ccnnnn cccBnn
cccBBBc nnnBB cccccccc cnnnnn Bcccc

are matched at least once, in part or whole, by the Flex regular expression

n[^a-z]B

(A) 15 (B) 2 (C) 4 (D) 1 (E) 3 (F) OTHER

nBBBn
nnnnnnn
ccccccc
BBBBBB
ncccc
nBB
cccccc
BBBnnBB
ccnnnn
cccBnn
cccBBBc
nnnBB
cccccccc
cnnnnn
Bcccc
ANS: C-4

Q 9.

How many of the following 15 strings

xxRv RRRRR vvvRRRR RRRRx Rxxvv Rxxv RRRvvRR vvvxRRR vxxR xRRRRRR
RRRvx RRRvvvxxx xxxxvv vvRRRR vvRxxx

are matched at least once, in part or whole, by the Flex regular expression

v[a-zA-Z]R

(A) 11 (B) 5 (C) 12 (D) 14 (E) 6 (F) OTHER

xxRv
RRRRR
vvvRRRR

RRRRx
Rxxvv
Rxxv
RRRvvRR
vvvxRRR
vxxR
xRRRRRR
RRRvx
RRRvvvxxx
xxxxvv
vvRRRR
vvRxxx
ANS: B-5

Q 10.

```
How many of the following 15 strings

jjjjjjjj RRRwRRR RRRRjjj wwjR wwwwwRRR RRRRRR wwww RRRww wwwRRRjj
jwww jjjjRR wwwRRjj Rjww wwjjjjjj jRRRw

are matched at least once, in part or whole, by the Flex regular expression

jj[a-zA-Z][a-zA-Z]+w

(A) 4 (B) 11 (C) 2 (D) 5 (E) 0 (F) OTHER
```

jjjjjjjj
RRRwRRR
RRRRjjj
wwjR
wwwwwRRR
RRRRRR
wwww
RRRww
wwwRRRjj
jwww
jjjjRR
wwwRRjj
Rjww
wwjjjjjj
jRRRw
ANS: E-0

Q 11.

```
How many of the following 15 strings

JJJeee egggggg JJJeegg ggggee geee ggggg eeeggge JJJgJJ eeJggg
eeeee JJJJJ JJJJJJ eeeeeg ggeeeJJJ JJJJ

are matched at least once, in part or whole, by the Flex regular expression

e[a-zA-Z]{2}gg

(A) 14 (B) 1 (C) 13 (D) 3 (E) 11 (F) OTHER
```

JJJeee
egggggg
JJJeegg
ggggee
geee
ggggg
eeeggge
JJJgJJ
eeJggg

eeeee
JJJJJ
JJJJJJ
eeeeeg
ggeeeJJJ
JJJJ
ANS: D-3

Q 12.

How many of the following 15 strings

nnntt nCCCC tCCCnnn nnttttt tCCCC nttnnn ttCC nnnCC tnnnnn CCC
CCCtttnn CttC CCCCttt CCtC CCCCt

are matched at least once, in part or whole, by the Flex regular expression

n[a-zA-Z]{1,2}tt

(A) 2 (B) 3 (C) 13 (D) 6 (E) 10 (F) OTHER

nnntt
nCCCC
tCCCnnn
nntttt
tCCCC
nttnnn
ttCC
nnnCC
tnnnnn
CCC
CCCtttnn
CttC
CCCCttt
CCtC
CCCCt
ANS: A-2

Q 13.

How many of the following 15 strings

xxxx ffxC fffCC fCCC CxCC ffCff CCxxf xxffxx xfx ffxff xxfff xCCff
fxxx xffff xCCx

are matched at least once, in part or whole, by the Flex regular expression

([A-Z]{2,3}|[a-z]{4})

(A) 12 (B) 13 (C) 7 (D) 9 (E) 4 (F) OTHER

xxxx
ffxC
fffCC
fCCC
CxCC
ffCff
CCxxf
xxffxx
xfx
ffxff
xxfff
xCCff
fxxx

xffff
xCCx
ANS: A-12

Q 14.

```
How many of the following 15 strings

JJJJ JJJkzzz kzk kkkJJJz zzzkkkk zzkkkkk zzJJJJ kkkkz JJkJJ kJzzz
kkkzzkk zzzzzkk JJJkkkJ JJJJJ kkkJJkk

are matched at least once, in part or whole, by the Flex regular expression

kk.

(A) 12 (B) 14 (C) 7 (D) 6 (E) 9 (F) OTHER
```

JJJJ
JJJkzzz
kzk
kkkJJJz
zzzkkkk
zzkkkkk
zzJJJJ
kkkkz
JJkJJ
kJzzz
kkkzzkk
zzzzzkk
JJJkkkJ
JJJJJ
kkkJJkk
ANS: C-7

Q 15.

```
How many of the following 15 strings

ddz zzd zzz zdE EzE EEd zEE zzE zEz dEz Edd EEE zdd Ezd Ezz

are matched at least once, in part or whole, by the Flex regular expression

^[a-z]

(A) 7 (B) 14 (C) 8 (D) 6 (E) 9 (F) OTHER
```

ddz
zzd
zzz
zdE
EzE
EEd
zEE
zzE
zEz
dEz
Edd
EEE
zdd
Ezd
Ezz
ANS: E-9

Q 16.

How many of the following 15 strings

oooo 00w00 00ww00 0oow wooo o00ww ooooww oo0w w0000 owwo oww00
0ww00 000w www oo0oo

are matched at least once, in part or whole, by the Flex regular expression

(o{2}|O{1,2}|[A-M]+)$

(A) 3 (B) 8 (C) 13 (D) 12 (E) 15 (F) OTHER

ooOO
OOwOO
OOwwOO
Ooow
wooo
oOOww
oooOww
ooOw
wOOOO
owwo
owwOO
OwwOO
OOOw
www
ooOoo
ANS: B-8

Q 17.

How many of the following 9 sentences

b bbbbbbbb zs3ZmXc bbbbbb bbbbb bb bbbb bbbbbbb sentence

are in the language defined by the Bison Context Free Grammar

```
%token b
%%
sentence: b | b sentence
;
```

(A) 5 (B) 9 (C) 6 (D) 7 (E) 1 (F) OTHER (3 marks)

b - ACCEPTED
bbbbbbbb - ACCEPTED
zs3ZmXc
bbbbbb - ACCEPTED
bbbbb - ACCEPTED
bb - ACCEPTED
bbbb - ACCEPTED
bbbbbbb - ACCEPTED
sentence
ANS: D-7

Q 18.

```
How many of the following 8 sentences

WWWWW WW WWWWWW WWWW WWWWWWW NSsg0EK W sentence

are in the language defined by the Bison Context Free Grammar

%token W
%%
sentence: W | sentence  W
;

(A) 1 (B) 6 (C) 5 (D) 3 (E) 4 (F) OTHER (3 marks)
```

WWWWW - ACCEPTED
WW - ACCEPTED
WWWWWW - ACCEPTED
WWWW - ACCEPTED
WWWWWWW - ACCEPTED
NSsg0EK
W - ACCEPTED
Sentence
ANS: B-6

Q 19.

```
How many of the following 7 sentences

sentence L3kcIUT jjjjFFFFFF jjjjjjFF jFFFFF jjjjjjjFFFFF jjjFF

are in the language defined by the Bison Context Free Grammar

%token j F
%%
sentence: sub | sub sentence
sub: j | F
;

(A) 3 (B) 1 (C) 7 (D) 5 (E) 2 (F) OTHER (3 marks)
```

sentence
L3kcIUT
jjjjFFFFFF - ACCEPTED
jjjjjjFF - ACCEPTED
jFFFFF - ACCEPTED
jjjjjjjFFFFF - ACCEPTED
JjjFF – ACCEPTED
ANS: D-5

Q 20.

How many of the following 10 sentences

dddEEEE ddE ddd ddddEEEE ddddE EE ddddEEE ddEE dddE dEEE

are in the language defined by the Bison Context Free Grammar

```
%token d E
%%
sentence: d | E | d sentence
;
```

(A) 4 (B) 3 (C) 8 (D) 5 (E) 9 (F) OTHER (3 marks)

dddEEEE
ddE - ACCEPTED
ddd - ACCEPTED
ddddEEEE
ddddE - ACCEPTED
EE
ddddEEE
ddEE
dddE - ACCEPTED
DEEE
ANS: A-4

Q 21.

How many of the following 10 sentences

rrrrRR rR rrRRR rrrR rrrrRRR rrrrRRRR rRRR rrrRR RR r

are in the language defined by the Bison Context Free Grammar

```
%token r R
%%
sentence: r | R | sentence r
;
```

(A) 7 (B) 4 (C) 2 (D) 10 (E) 1 (F) OTHER (3 marks)

rrrrRR
rR
rrRRR
rrrR
rrrrRRR
rrrrRRRR
rRRR
rrrRR
RR
r- ACCEPTED
ANS: E-1

Q 22.

```
How many of the following 10 sentences

xxxMMMM xxx xxM xM MMM xxxxMMMM xMM xxMMM xxxxMMM xxMM

are in the language defined by the Bison Context Free Grammar

%token x M
%%
sentence: x | M | M sentence
;

(A) 4 (B) 8 (C) 1 (D) 9 (E) 7 (F) OTHER (3 marks)
```

xxxMMMM
xxx
xxM
xM
MMM - ACCEPTED
xxxxMMMM
xMM
xxMMM
xxxxMMM
xxMM
ACCEPTED: C-1

Q 23.

```
How many of the following 10 sentences

xxVV VV xxVVVV xxxxVVV xVV xxxx xxVVV xxxxVV xxxV xxxVVVV

are in the language defined by the Bison Context Free Grammar

%token x V
%%
sentence: x | V | sentence V
;

(A) 2 (B) 4 (C) 9 (D) 1 (E) 5 (F) OTHER (3 marks)
```

xxVV
VV - ACCEPTED
xxVVVV
xxxxVVV
xVV - ACCEPTED
xxxx
xxVVV
xxxxVV
xxxV
XxxVVVV
ANS: A-2

Q 24.

How many of the following 5 sentences

uuuuuuuuuuu uuuu;u;uuu; uuuu;u;uu;u; u;uuuuu;uuu; uuu;uuuuuu;

are in the language defined by the Bison Context Free Grammar

```
%token u
%%
sentence: list | sentence list
list: listc ';'
listc: u | u listc
;
```

(A) 1 (B) 5 (C) 3 (D) 4 (E) 2 (F) OTHER (3 marks)

uuuuuuuuuuu
uuuu;u;uuu; - ACCEPTED
uuuu;u;uu;u; - ACCEPTED
u;uuuuu;uuu; - ACCEPTED
uuu;uuuuuu; - ACCEPTED
ANS: D-4

Q 25.

How many of the following 7 sentences

qqqqq q,qq,q,q qq,q,q q,qq,q,q, qqqq,q, qq,q,q qqqq

are in the language defined by the Bison Context Free Grammar

```
%token q
%%
sentence: listc | listc ',' sentence
listc: q | q listc
;
```

(A) 7 (B) 6 (C) 4 (D) 5 (E) 2 (F) OTHER (3 marks)

qqqqq - ACCEPTED
q,qq,q,q - ACCEPTED
qq,q,q - ACCEPTED
q,qq,q,q,
qqqq,q,
qq,q,q - ACCEPTED
qqqq – ACCEPTED
ANS: D-5

Q 26.

How many of the following 5 sentences

LL,LL,; L,LLL,L,; L,LL; LL,L,L, LLL,L

are in the language defined by the Bison Context Free Grammar

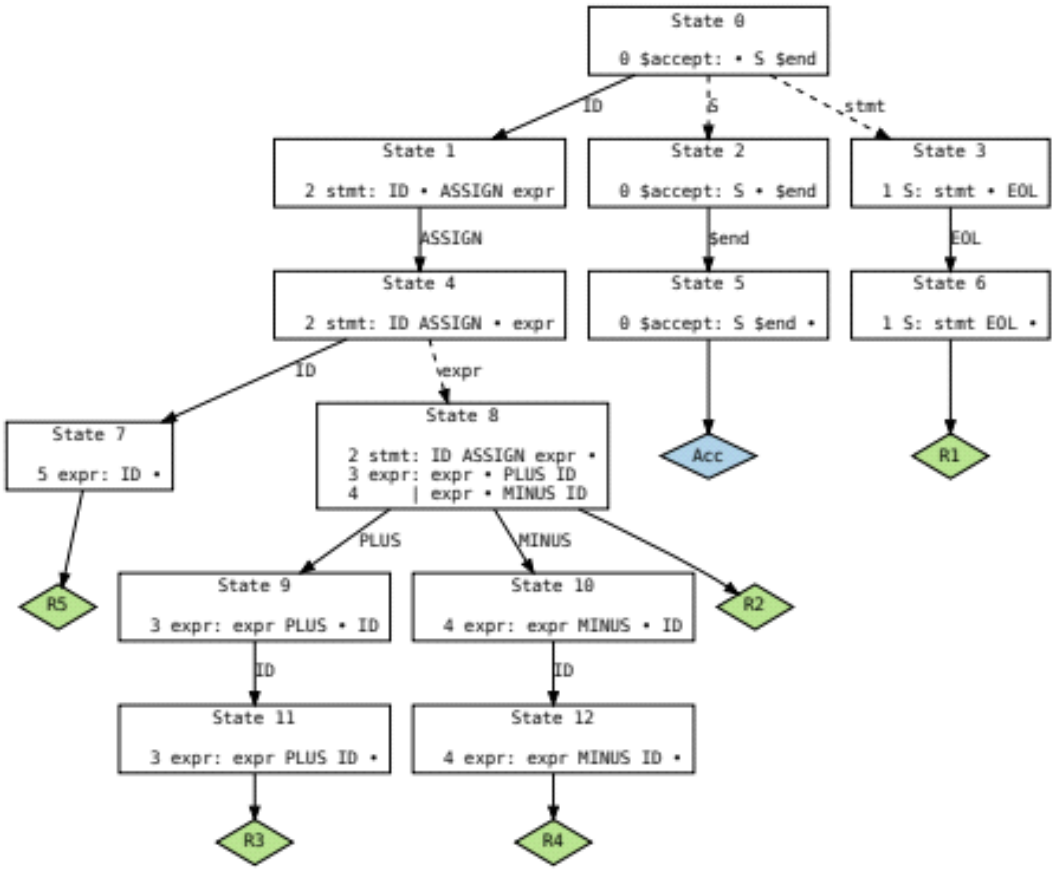
```
%token L
%%
sentence: commal ';'
commal: listc | listc ',' commal
listc: L | L listc
;
```

(A) 5 (B) 1 (C) 2 (D) 4 (E) 3 (F) OTHER (3 marks)

LL,LL,;
L,LLL,L,;
L,LL; - ACCEPTED
LL,L,L,
LLL,L
ANS: B-1

Q 27.

```
Given the following tokens
"+"     { return PLUS; }
"-"     { return MINUS; }
":="    { return ASSIGN; }
[a-z]   { yylval = yytext[0]; return ID; }
\n      { return EOL; }
and the following Bison Context Free Grammar
    0 $accept: S $end
    1 S: stmt EOL
    2 stmt: ID ASSIGN expr
    3 expr: expr PLUS ID
    4     | expr MINUS ID
    5     | ID
which generates the Bison Shift Reduce Parser
```




```
What sequence of states will the Bison Shift Reduce Parser go through
parsing the sentence
g:=a+b+++\n
 (22 marks)
```

0: Start in state 0. input token "g". "g" is an ID. Shift to state 1.
The stack has [0, 1 – ID].

1: input token ":=". ":=" is an ASSIGN. Shift to state 4.

The stack has [0, 1 – ID, 4 – ASSIGN].

4: input token "a". "a" is an ID. Shift to state 7.
The stack has [0, 1-ID, 4-ASSIGN, 7-ID].

7: input token "+". Cannot be consumed in this state.
Reduce according to rule five.
Rule five says that can expr can be generated from an ID.
The top of the stack reads [7-ID].
We can pop this off the stack.
From state 4 and the expr we have just generated, we can move to state 8.
The stack has [0, 1-ID, 4-ASSIGN, 8-expr].

8: input token "+". "+" is a PLUS. Shift to state 9.
The stack has [0, 1-ID, 4-ASSIGN, 8-expr, 9-PLUS].

9: input token "b". "b" is an ID. Shift to state 11.
The stack has [0, 1-ID, 4-ASSIGN, 8-expr, 9-PLUS, 11-ID].

11: input token "+". Cannot be consumed in this state.
Reduce according to rule three.
Rule three says that expr can be generated from expr, PLUS, ID.
The top of the stack currently has [8-expr, 9-PLUS, 11-ID].
Pop this from the stack. The stack has [0, 1-ID, 4-ASSIGN].
From state 4 and the expr we have just generated, we can move to state 8.
The stack has [0, 1-ID, 4-ASSIGN, 8-expr].

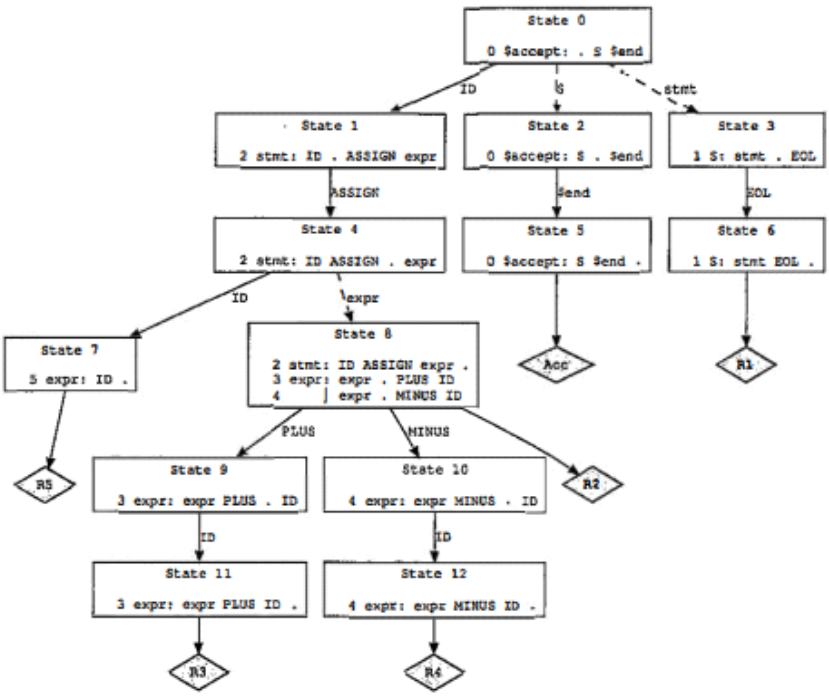
8: input token "+". "+" is a PLUS. Shift to state 9.
The stack has [0, 1-ID, 4-ASSIGN, 8-expr, 9-PLUS].

9: input token "+". Cannot be consumed in this state.
Cannot reduce according to rule three in state nine.
Cannot shift to any other state.
This string "g:=a+b+++\n" is not valid to the given grammar.
The Bison shift-reduce parser cannot parse the expression.

2020

Monday, April 28, 2025 10:51 AM

```
Given the following tokens
"+"     { return PLUS; }
"-"     { return MINUS; }
":="    { return ASSIGN; }
[a-z]   { yylval = yytext[0]; return ID; }
\n      { return EOL; }
and the following Bison Context Free Grammar
    0 $accept: S $end
    1 S: stmt EOL
    2 stmt: ID ASSIGN expr
    3 expr: expr PLUS ID
    4     | expr MINUS ID
    5     | ID
which generates the Bison Shift Reduce Parser
```




```
What sequence of states will the Bison Shift Reduce Parser go through
parsing the sentence
w:=b+c-a\n
```

0: input symbol "w". "w" is an ID. Shift to state 1.
The stack has [0, 1-ID]

1: input symbol ":=". ":=" is an ASSIGN. Shift to state 4.
The stack has [0, 1-ID, 4-ASSIGN]

4: Input symbol "b". "b" is an ID. Shift to state 7.
The stack has [0, 1-ID, 4-ASSIGN, 7-ID]

7: Input symbol "+". Input symbol cannot be consumed in this state.
Reduce according to rule five. There is an ID on top of the stack.
Pop this off the stack.
From state 4 and the expr we have just generated, we can move to state 8
The stack has [0, 1-ID, 4-ASSIGN, 8-expr]

8: input symbol "+". "+" is a PLUS. Shift to state 9.
The stack has [0, 1-ID, 4-ASSIGN, 8-expr, 9-PLUS]

9: Input symbol "c". "c" is an ID. Shift to state 11.
The stack has [0, 1-ID, 4-ASSIGN, 8-expr, 9-PLUS, 11-ID]

11: Input symbol "-". Input symbol cannot be consumed in this state.
Reduce according to rule three. There is [expr, PLUS, ID] on top of the stack.
Pop this off the stack.
From state 4 and the expr we have just generated, we can move to state 8.
The stack has [0, 1-ID, 4-ASSIGN, 8-expr].

8: input symbol "-". "-" is a MINUS. Shift to state 10.
The stack has [0, 1-ID, 4-ASSIGN, 8-expr, 10-MINUS]

10: Input symbol "a". "a" is an ID. Shift to state 12.
The stack has [0, 1-ID, 4-ASSIGN, 8-expr, 10-MINUS, 12-ID]

12: Input symbol "\n". Input symbol cannot be consumed in this state.
Reduce according to rule four. There is [expr, MINUS, ID] on top of the stack.
Pop this off the stack.
From state 4 and the expr we have just generated, we can move to state 8.
The stack has [0, 1-ID, 4-ASSIGN, 8-expr]

8: Input symbol "\n". Input symbol cannot be consumed in this state.
Reduce according to rule two. There is [ID, ASSIGN, expr] on top of the stack.
Pop this off the stack.
From state 0 and the stmt we have just generated, we can move to state 3.

3: input symbol "\n". "\n" is an EOL. Shift to state 6.
The stack has [0, 3-stmt, 6-EOL]

6: Reduce according to rule one. There is [stmt, EOL] on top of the stack.
Pop this off the stack.
From state 0 and the S we have just generated, we can move to state 2.

2: from state 2, we have a parsed string that can be accepted by the rules of the language.
Shift to the accepting state.

5: The accepting state.