HW 2

3.3.2

Describe the languages denoted by the following regular expressions

a

$a(a|b) * a$

> ✓ **Answer**
> "a" followed by any amount of "a"s or "b"s, then ending surely with an "a"

b

$((\epsilon|a)b*)*$

> ✓ **Answer**
> A combination of "a"s and "b"s in any order or repetition. Could be completely empty.

c

$(a|b) * a(a|b)(a|b)$

> ✓ **Answer**
> An "a" that is preceeded with any amounts of "a"s and "b"s, and followed by "a" or "b" and "a" or "b".

d

$a * ba * ba * ba*$

> ✓ **Answer**
> A string of "a"s and "b"s where you can have any amount of "a"s (including 0), and must have exactly 3 "b"s dispersed among the string.

e

$(aa|bb) * ((ab|ba)(aa|bb) * (ab|ba)(aa|bb)*)*$

✓ **Answer**

A string of "a"s and "b"s that has an even length, and any combination of "a"s or "b"s among it.

3.3.5

Write regular definitions for the following languages

c

Comments, consisting of a string surrounded by `/*` and `*/`, without an intervening `*/`, unless it is inside double-quotes (`"`).

✓ **Answer**

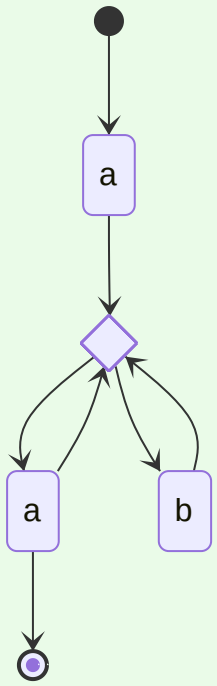
`(\/\*)(("[^"]*")*[^"^(\*\/)])*\*\/`

3.4.1

Provide transition diagrams to recognize the same languages as each of the regular expressions in Exercise 3.3.2.

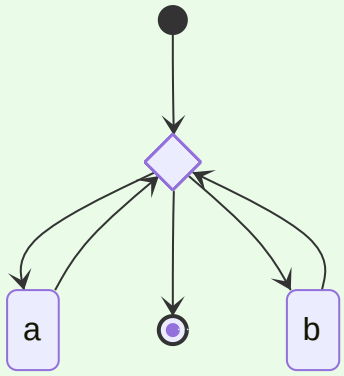
a

✓ **Answer**



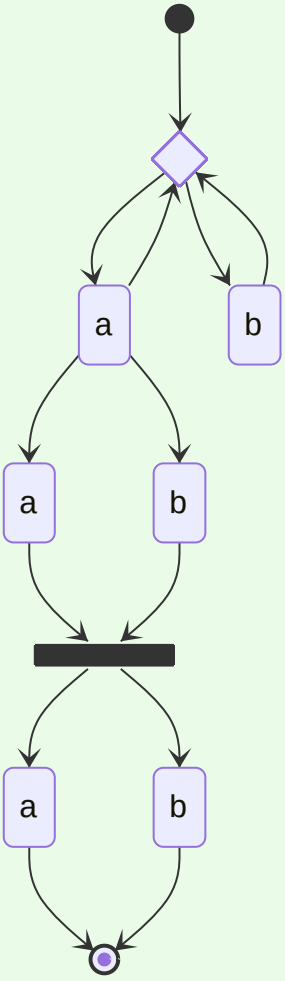
b

✓ **Answer**



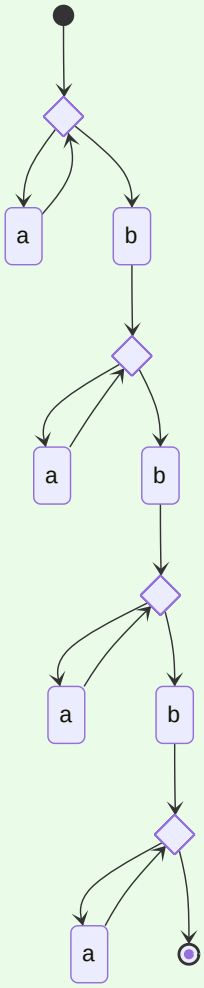
c

✓ **Answer**



d

✓ **Answer**

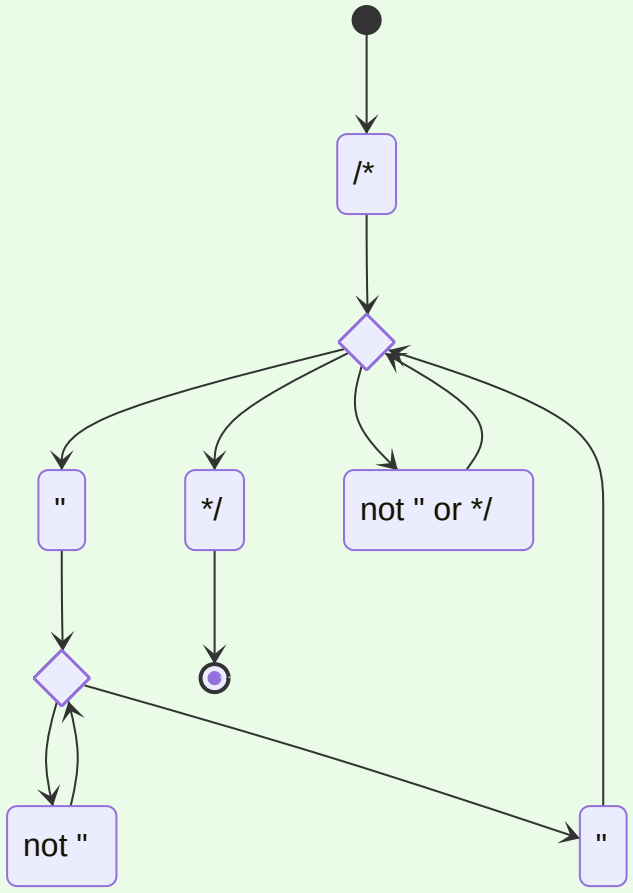


3.4.2

Provide transition diagrams to recognize the same languages as each of the regular expressions in Exercise 3.3.5.

c

✓ **Answer**



3.6.2

c

✓ **Answer**

An NFA for the graph given in the previous question would be as follows:

State	Description	ϵ	$''$	$*/$	$/*$	$other$
Start	Start	$\emptyset$	$\emptyset$	$\{0\}$	$\emptyset$	$\emptyset$
0	`/*`	$\{1\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
1	Loop Point	$\emptyset$	$\{2\}$	$\{3\}$	$\{4\}$	$\{4\}$
2	`"`	$\{5\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
3	`*/`	$\{End\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
4	Not `"` or `*/`	$\{4\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
5	Loop Point	$\emptyset$	$\{7\}$	$\{6\}$	$\{6\}$	$\{6\}$
6	Not `"`	$\{5\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
7	`"`	$\{1\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

State	Description	ϵ	$''$	$*/$	$/*$	$other$
End	End	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

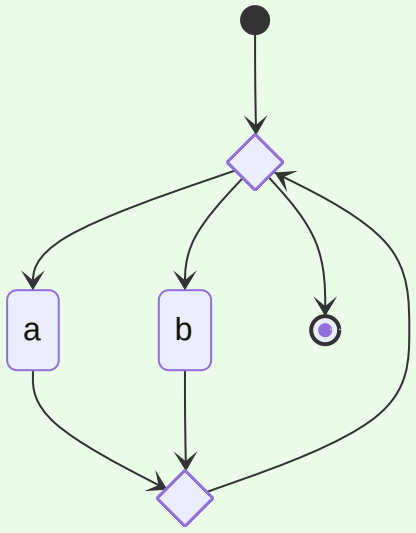
3.7.3

a

$(a|b)*$

✓ **Answer**

First, we utilize algorithm 3.23 to produce this graph:



From the graph, we get the following transition table:

State	Description	ϵ	a	b
Start	Start	$\{0\}$	$\emptyset$	$\emptyset$
0	Loop Start	$\{End\}$	$\{1\}$	$\{2\}$
1	`a`	$\{3\}$	$\emptyset$	$\emptyset$
2	`b`	$\{3\}$	$\emptyset$	$\emptyset$
3	Loop End	$\{0\}$	$\emptyset$	$\emptyset$

State	Description	ϵ	a	b
End	End	$\emptyset$	$\emptyset$	$\emptyset$

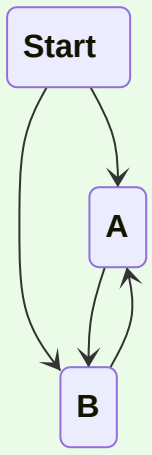
Applying algorithm 3.20 provides us the following DFA table:

NFA State	DFA State	a	b	End
$\{Start, 0, End\}$	α	A	B	Yes
$\{1, 3, 0, End\}$	A	A	B	Yes
$\{2, 3, 0, End\}$	B	A	B	Yes

Providing just the transition table gives:

State	a	b	End
Start	A	B	Yes
A	A	B	Yes
B	A	B	Yes

Finally, drawing the final diagram provides:



Compiler

Information about our compiler may be found in the following GitHub repo:

https://github.com/404Wolf/typescripten/

Test runs for the textbook example produce the following result:

https://github.com/404Wolf/typescripten/blob/main/textbook-run.ans

```
{ basic id ; basic id ; basic id ; basic id ; basic [ num ] id ; while (
true ) { do id = id + num ; while ( id [ id ] < id ) ; do id = id - num
; while ( id [ id ] > id ) ; if ( id ⩾ id ) break ; id = id [ id ] ; id
[ id ] = id [ id ] ; id [ id ] = id } }

Parsing completed successfully.

Symbol Table:
————
ID: j
ID: i
ID: x
ID: a
ID: v
————
```

Docker containers containing the application are bundled with the submission and can be run with:

```
docker load < result
cat input.txt | docker run localhost/compiler:latest
```

Source can be found on the GitHub provided earlier and attached with this assignment.