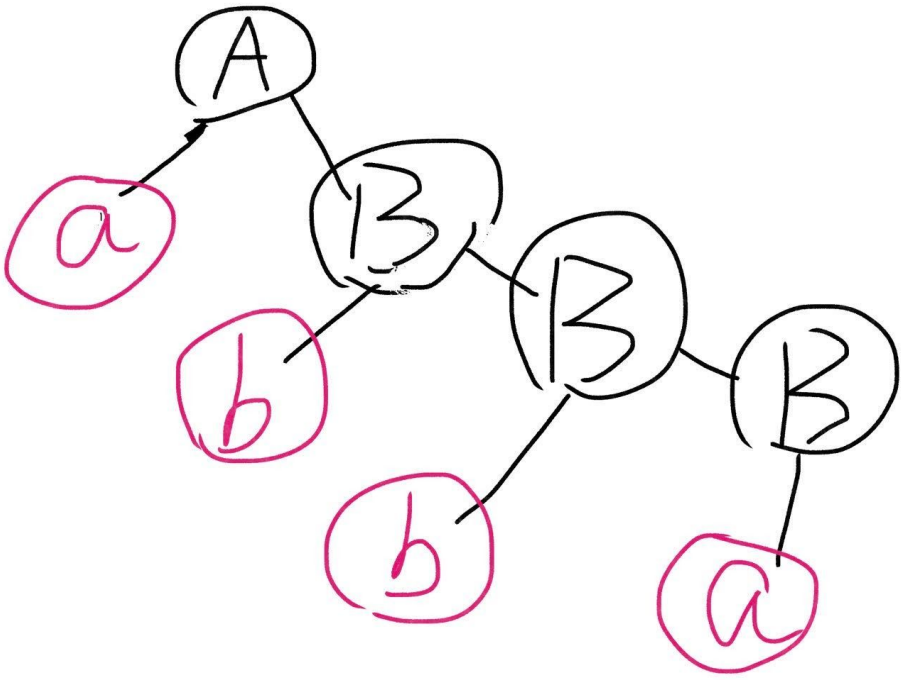
1. Suppose you have a lot of files and want to check whether they have valid names or not. A valid name starts with "file" and ends with ".pdf" but there must be more characters (ie. "file.pdf" is not allowed). Give a regular expression which would check the filenames.

file.+.pdf

2. draw the (concrete) parse tree generated by a naive top-down recursive parser, for the string "abba" under the following grammar rules:

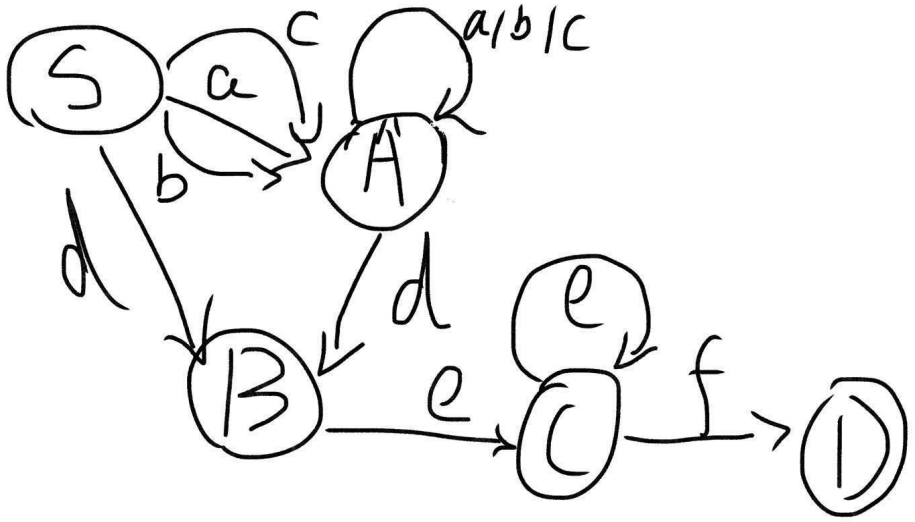
```
A ::= a B
B ::= b B | a
```



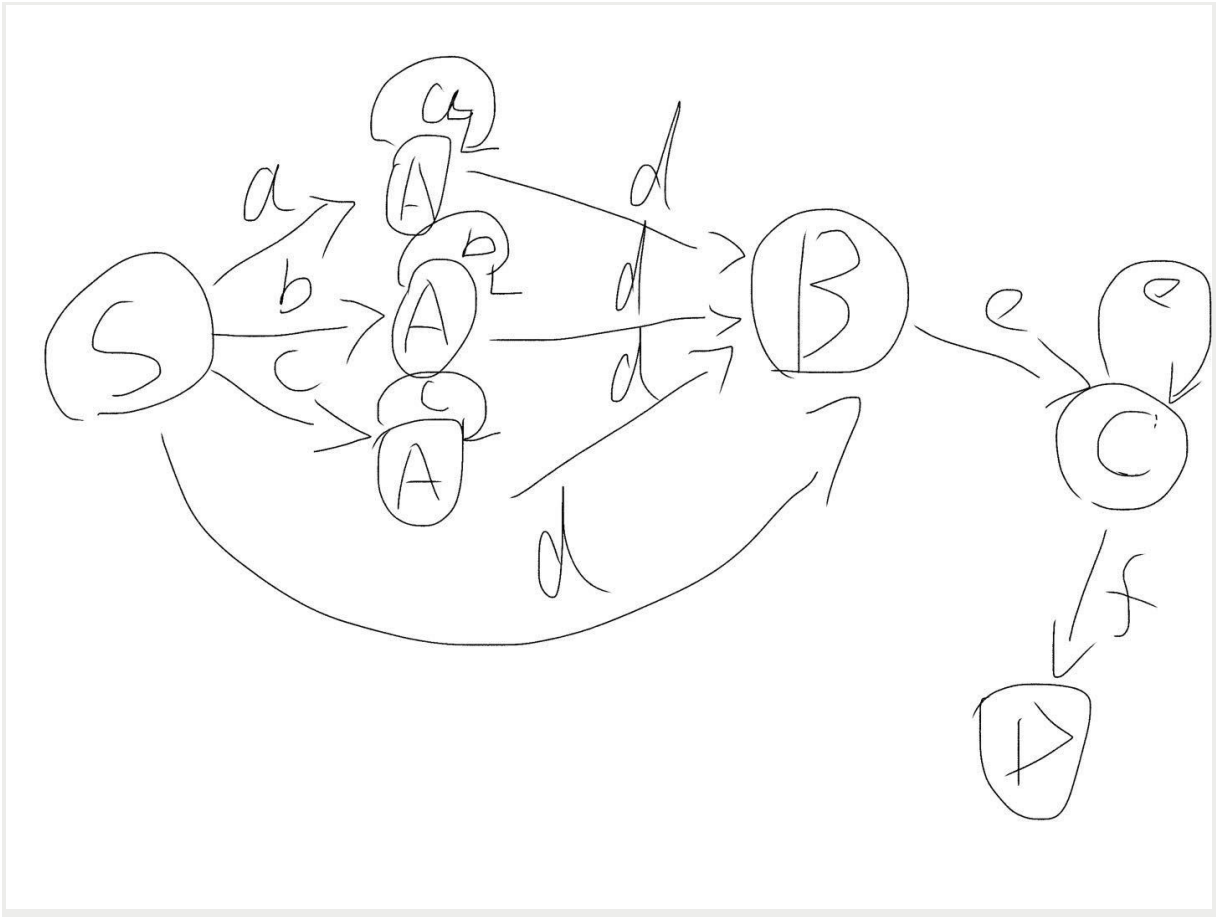

3. describe in one sentence the strings accepted by the regex `(ab)+c*`

ab as a capturing group has one or more copies followed by 0 or more copies of c

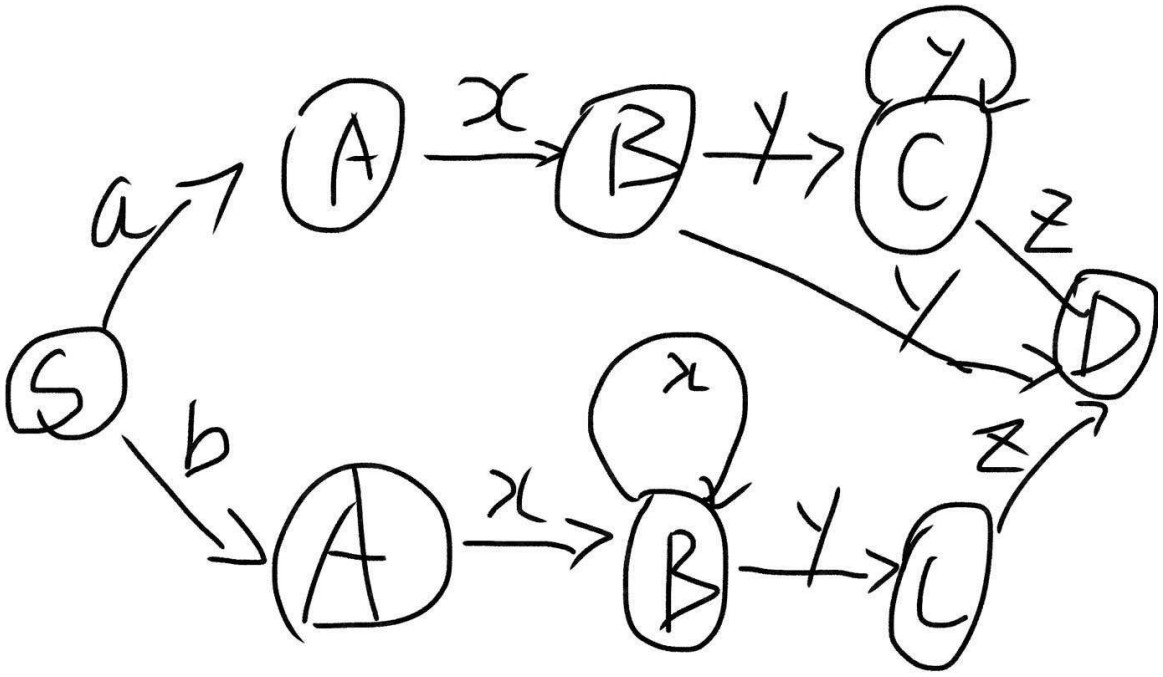
4. draw the Acceptor automaton for the regex `(a|b|c)*d(e+)f`




or:



5. draw the Acceptor automaton for the regex `(axy*z)|(bx+yz)`




6. Which ones of the sentences *(a)* to *(g)* given below belong to the language defined by the following grammar?

(a) (b) (d)(f)

```
MUD ::= MAD CHOP
CHOP ::= "stop" DOG | DOG "run" | "wet" DOG
MAD ::= [e-z0-9]+ | DOG
DOG ::= (yeh\s)*(nah\s)*(bro)+
```

| | sentence |
|---|---|
| *(a)* | `yeh nah bro stop bro` |
| *(b)* | `bro bro bro bro bro run` |
| *(c)* | `yeh bro stop yeh yeh yeh nah nah nah` |
| *(d)* | `yeh09 wet bro` |
| *(e)* | `ex wet yeh nah` |
| *(f)* | `one1 wet bro` |
| *(g)* | `2020 dog run` |

Consider the following (partial) grammar rules:

```
T  ::= NAMEOF | T "with" T | T "without" T | "(ie." T ")"
NAMEOF ::=
```

7. Complete the definition of `NAMEOF` using a regular expression (regex) so that it describes an uppercase letter followed by one or more lowercase letters.
NAMEOF ::= [A-Z] [a-z]+
8. Change the grammar (from question 7) so that it is now LL1 (but still defines the same language).
T ::= NAMEOF | T "with" T | T "without" T | "(ie." T ")"

NAMEOF ::=[A-Z]([A-Z]| [a-z])+ [0-9]*

9. Change the following grammar so that it can be parsed by a naive top-down recursive descent parser:

```
LIST ::= NUM | LIST "," NUM
NUM ::= [0-9]+
```

LIST ::= NUM | GET "," NUM
GET ::= “GET” “,” LIST
NUM ::= [0-9]+