Algorithms - HW4

Ans 1-

a) x^*

$S \rightarrow \epsilon | xS$

b) z^+

$S \rightarrow z | zS$

c) $a|b$

$S \rightarrow a | b$

d) 01

$S \rightarrow 0A$

$A \rightarrow 1$

e) $(01)^*$

$S \rightarrow 0A | \epsilon$

$A \rightarrow 1S$

f) $(01)^* | 1$

$S \rightarrow 0A | \epsilon | 1$

$A \rightarrow 1B$

$B \rightarrow 0A | \epsilon$

Ans 2-

a) $S \rightarrow \epsilon | xS$

$S \Rightarrow xS \Rightarrow xxS \Rightarrow xx$

b) $S \rightarrow aB | \epsilon$

$B \rightarrow bS$

$S \Rightarrow aB \Rightarrow abS \Rightarrow abaB \Rightarrow ababS \Rightarrow abab$

c) $S \rightarrow 0B$

$B \rightarrow 1B \mid 0C$

$C \rightarrow \epsilon$

$S \Rightarrow 0B \Rightarrow 01B \Rightarrow 011B \Rightarrow 0111B \Rightarrow 01110C \Rightarrow 01110$

Ans 3-

a) $S \rightarrow aSa \mid bSb \mid cSc \mid \epsilon$

$S \Rightarrow aSa \Rightarrow abSba \Rightarrow abbSbba \Rightarrow abbcScbba \Rightarrow abbcbSbcbba$

$\Rightarrow abbcbbcbba$

b) $S \rightarrow XSX \mid \epsilon$

$X \rightarrow a \mid b \mid c$

$S \Rightarrow XSX \Rightarrow XXSXX \Rightarrow aXSXX \Rightarrow abSXX \Rightarrow abXX \Rightarrow abcX \Rightarrow abca$

$S \Rightarrow XSX \Rightarrow aSX \Rightarrow aXSXX \Rightarrow abSXX \Rightarrow abXX \Rightarrow abcX \Rightarrow abca$

Ans 4-

<hostport> ::= <host><exthost>

<exthost> ::= ":"<port> | " "

<port> ::= <digits>

<digits> ::= <digit> | <digit><digits>

<digit> ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

<hpath> ::= <hsegment><ehsegment>

<ehsegment> ::= "/"<hpath> | " "

<hsegment> ::= " " | <uchar><hsegment> | ";"<hsegment> |
":"<hsegment> | "@"<hsegment> | "&"<hsegment> |
"="<hsegment>

<search> ::= " " | <uchar><search> | ";"<search> |
":"<search> | "@"<search> | "&"<search> | "="<search>

Ans 5-

a) <syntax> is the start rule for parsing BNF.

b) Non-terminal must begin with an opening angle bracket (<) without small brackets
First letter of the non-terminal must begin with capital OR small letter.

c) After the first character, a rule name can have letters, digits OR "-"

d) Rule name can have infinitely many characters. They should have a minimum of one character.

e) Literals are a set of characters (letters, symbols and digits). These can be written inside single quotation ' ' OR can be written inside " " quotation marks.

f) "\"" OR '"'

g) "\"'" OR '"'"'"'

Ans 6-

a) In parsing expression grammers, choice operator is ordered (unlike BNF where any of the choices from the choice operator can be chosen). If the first alternative succeeds, the second alternative is ignored. Hence, in this case it is always the first "if" then "else" ~~if~~ expression that will be matched. If this match fails, only then will the second expression be matched.
Hence, with PEG's, we do not ~~care about~~ have a problem if alternatives have the same prefix.

b) S = ("if" C "then" S "else" S) / ("if" C "then" S)

S ⇒ "if" C "then" S "else" S
⇒ "if" (test1) "then" S "else" S
⇒ "if" (test1) "then" "if" C "then" S "else" S
⇒ "if" (test1) "then" "if" (test2) "then" S "else" S
⇒ "if" (test1) "then" "if" (test2) "then" print(3) "else" S
⇒ "if" (test1) "then" "if" (test2) "then" print(3) "else" print(0)

c) The start symbol 'S' in the first line has the 'else' clause. When the derivation starts, the first step involves :

S ⇒ "if" C "then" S "else" S

All citations :

i) https://en.wikipedia.org/wiki/Regular-grammer
ii) https://en.wikipedia.org/wiki/Backus%E2%80%93Naur-form
iii) https://tools.ietf.org/html/rfc1738
iv) https://en.wikipedia.org/wiki/Parsing_expression-grammer
v) baturin.org/tools/bufgen
vi) w3.org/Notation.html