b)

i)

re ::= re ‘|’ union | union

union ::= union ‘&’ intersect | intersect

intersect ::= intersect not | not

not ::= ‘~’ star | star

star ::= atom ‘+’ | atom ‘\*’ | atom ‘?’ | atom

atom ::= ‘!’ | ‘#’ | ‘.’ | ‘c’

ii) Recursive descent parse: top to bottom, left to right.

Going from top to bottom and left to right will force the recursion to always match on the non-terminal on the left hand side and continue infinitely rather than reaching a terminal in the grammar. For the parsing to work the production must be consuming the characters from the string to match a terminal, but in this case the left recursion will continue to call the function corresponding to a non-terminal to try to match that non-terminal resulting in the infinite loop.

iii)

re ::= union

union ::= intersect unions

unions ::= empty | ‘|’ intersect unions

intersect ::= concat intersects

intersects ::= empty | ‘&’ concat intersects

concat ::= not concats

concats ::= empty | not concats

not ::= ‘~’ not | star

star ::= atom stars

stars ::= empty | ‘\*’ atom stars | ‘+’ atom stars | ‘?’ atom stars

atom ::= ‘!’ | ‘#’ | ‘.’ | ‘c’