Will Christie
CSCI 3155
Lab 6 Write Up

2. b) i.
Based on precedence of operators: (union = lowest precedence)

re ::= union
union ::= union ‘|’ intersect | intersect
intersect ::= intersect ‘&’ concat | concat
concat ::= concat not | not
not ::= ‘~’ not | star
star ::= star ‘*’ | star ‘+’ | star ‘?’ | atom
atom ::= ‘!’ | ‘#’ | c | ‘.’ | ‘(‘ re ‘)’

2. b) ii.
This would do into an infinite loop because we are not able to recurse on progressively smaller problems. It sees one character, and has no way of knowing whether it will need to allow for more of the same matches (repeats), or whether it is going to settle on the next non-terminal. This results in recursing on the same non-terminal match (which is in fact itself) because it is unable to determine if it has allowed for enough characters in its match.

Non-regex example:
E ::= x | E + x

In the example above, we are unable to determine if the expression ‘x’ isn’t ‘x+x’ or ‘x+x+x’ based on the first character match. Because of this we are unable to let E just be x and because we cannot be sure that the expression is not just ‘x’, we are unable to settle on letting E be E+x as this makes the expression much too long. Ultimately, we enter an infinite recursion on E trying to figure out what it is supposed to be (which we can’t figure our until the expression is parsed, which won’t happen until we settle on E).

2. b) iii.
re :: = union
union ::= intersect unions
unions ::= **ε** | ‘|’ intersect unions
intersect ::= concat intersects
intersects ::= **ε** | ‘&’ concat intersects
concat ::= not concats
concats ::= **ε** | not concats
not ::= ‘~’ not | star
star ::= atom stars
stars ::= **ε** | ‘*’ stars | ‘+’ stars | ‘?’ stars
atom ::= ‘!’ | ‘#’ | c | ‘.’ | ‘(‘ re ‘)’

2. c) i.

TYPEREGEX

$$\frac{}{\Gamma \vdash /\text{\^{}re\$}/ : \text{RegExp}}$$

TYPECALLTEST

$$\frac{\Gamma \vdash e1 : \text{RegExp} \quad \Gamma \vdash e2 : \text{string}}{\Gamma \vdash e1.\text{test}(e2) : \text{bool}}$$

SEARCHCALLREGEX1

$$\frac{e1 \rightarrow e1'}{e1.f\,(e2) \rightarrow e1'.f\,(e2)}$$

SEARHCALLREGEX2

$$\frac{r = /\text{\^{}re\$}/ \quad e2 \rightarrow e2'}{r.f\,(e2) \rightarrow r.f\,(\,e2'\,)}$$

DOCALLREGEX

$$\frac{r = /\text{\^{}re\$}/ \quad s = str \quad b' = \text{test}(r, s)}{r.\text{“test”}(s) \rightarrow b'}$$