Login:

1. [6 points] Consider a really simplified set of regular expressions (call them RSREs) consisting of letters 'a' and 'b', concatenation, union (using the symbol '|'), epsilon (represented by the letter e), and Kleene star, but without parentheses. So, for example,

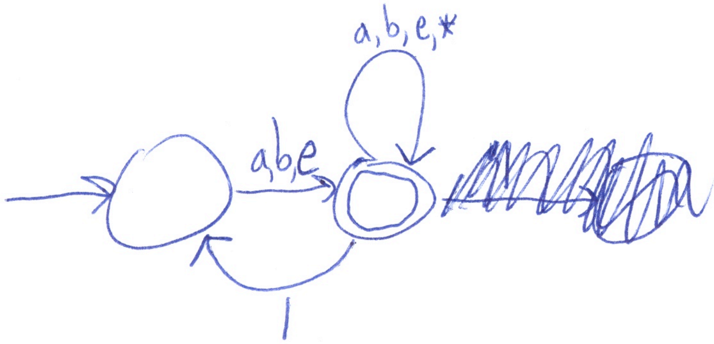
`a` `ab` `a*b|b|ba`

are valid RSREs.

a. Write a regular expression that recognizes RSREs. Use just the basic operators supplied by Flex or Jflex.

[abe][abe*]* ('|' [abe][abe*]*) *

b. Draw a DFA that recognizes RSREs.



2. [6 points] Now consider a pretty simple set of regular expressions (we'll call them PSREs) that are just like RSREs from the first problem except that they also allow parentheses for grouping. As usual, when not grouped by parentheses, Kleene star has highest precedence, followed by concatenation, and finally by union.

a. Write an *unambiguous* grammar for PSREs. Be sure to observe precedence properly, or you'll have trouble with part b.

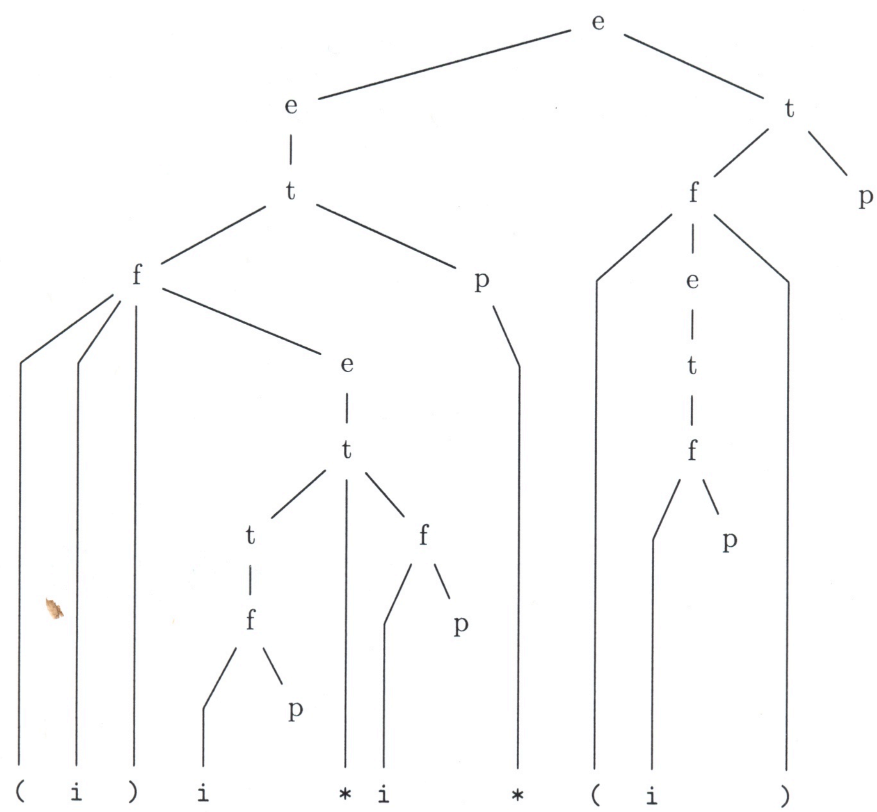
b. Write actions for the grammar of part (a) so that the semantic value you assign to the start symbol is length of the shortest string recognized by the regular expression being parsed. For example, for the PSRE '`ab`', the actions of your grammar should compute the value 2; for '`a|bb`' compute 1; for '`e`' and for '`(ab)*|c`', compute 0.

Part a	**Part b**
S = term '|' S	{ $$ = min ~~max~~($term, $S); }
| term	{ $$ = $term; }
term = prim term	{ $$ = $prim + $term; }
| prim	{ $$ = $prim; }
prim = prim '*'	{ $$ = 0; }
| '(' S ')'	{ $$ = $S; }
| a	{ $$ = 1; }
| b	{ $$ = 1; }
| e	{ $$ = 0; }

3. [1 point] I noticed that a certain kind of prepackaged snack sold in our local coffee shops has the slogan "The ones you remember," which I thought was mildly clever. Why?

Because the snack is a madeleine.

4. [8 points] Consider the parse tree below. All terminal symbols are written at the same level on the bottom row.



a. Show as much of the grammar as one can deduce from this parse tree.

e = t | e t

t = f | f p | t '*' f

f = 'i' p | '(' e ')' | '(' 'i' ')' e

p = ϵ | '*'

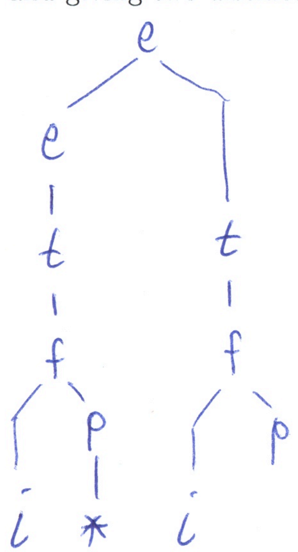
More questions on the next page

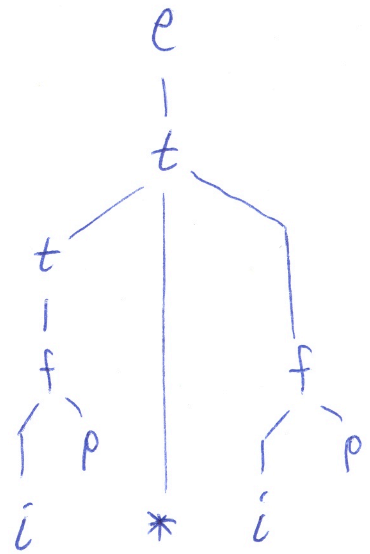
b. Show a reverse rightmost derivation corresponding to this tree, as performed by a shift-reduce parser. (However, you need not mention the shifts and reductions, just the sentential forms that get you back to the start symbol from the input).

`( i ) i * i * ( i )` →

(i) i p * i * (i)
(i) f * i * (i)
(i) t * i * (i)
(i) t * i p * (i)
(i) t * f * (i)
(i) t * (i)
(i) e * (i)
f * (i)
f p (i)
t (i)
e (i)
e (i p)
e (f)
e (t)
e (e)
f
f p
t
e

c. Show that the grammar is ambiguous by giving a string that has (at least) two parses and giving two distinct parse trees for it.





Continued on next page.

d. Given one of the parse trees for an ambiguous string in this language, can you find two leftmost derivations that give rise to that parse tree? If so, show such a tree and show the derivations; otherwise say why you cannot.

No – a ~~the~~ parse tree is uniquely defined by its leftmost derivation.