ASSIGNMENT-2 COMPILERS

(Itish Agarwal, 18CS30021

22 Sep' 2020)

Q1. (a) Let α be set of all lowercase letters except vowels, ie,

$\alpha = \{ b, c, d, f, g, h, j, k, l, m, n, p, q, r, s, t, v, w, x, y, z \}$

Then, required regex =

$$\alpha^* a(a|\alpha)^* e(e|\alpha)^* i(i|\alpha)^* o(o|\alpha)^* u(u|\alpha)^*$$

(b) Regular expression for strings containing even number of a's and odd number of b's:

$$(b|a(bb|aa)^*(ba|ab))(bb|aa|(ba|ab)(bb|aa)^*(ba|ab))^* + b(bb)^*$$

(c) Required regex =

$$b^*(a|ab)^*$$

(d) Required regex =

$$b^*a^*(\epsilon|b)a^*$$

Q2. /* Regular expression definitions */

```
INT     "int"
FLT     "float"
DBL     "double"
ID      [a-zA-Z][a-zA-Z0-9]*
RTN     "return"
WS      [ \t\n]
PUNC    [;]
%%

/* Transition Rules */

{INT}     {printf("<KEYWORD, int> \n");}
{FLT}     {printf("<KEYWORD, float> \n");}
```

```
{DBL}     { printf ("<KEYWORD, double> \n"); }
{RTN}     { printf ("<KEYWORD, return> \n"); }
{ID}      { printf ("<ID, %s> \n", yytext); }
"+"       { printf ("<OPERATOR, +> \n"); }
"^"       { printf ("<OPERATOR, ^> \n"); }
"="       { printf ("<OPERATOR, => \n"); }
"+="      { printf ("<OPERATOR, +=> \n"); }
"/="      { printf ("<OPERATOR, /=> \n"); }
"{"       { printf ("<SPECIAL SYMBOL, {> \n"); }
"}"       { printf ("<SPECIAL SYMBOL, }> \n"); }
","       { printf ("<SPECIAL SYMBOL, ,> \n"); }
"("       { printf ("<SPECIAL SYMBOL, (> \n"); }
")"       { printf ("<SPECIAL SYMBOL, )> \n"); }
{PUNC}    { printf ("<PUNCTUATION, ;> \n"); }
{WS}      /* white-space Rule */

%%
```

Token stream for given input:

<KEYWORD, float> <ID, Function2Calculate>
<SPECIAL SYMBOL, (><KEYWORD, int> <ID, a>
<SPECIAL SYMBOL, ,><KEYWORD, double> <ID, b>
<SPECIAL SYMBOL, ,> <KEYWORD, float> <ID, c>
<SPECIAL SYMBOL,)> <SPECIAL SYMBOL, {>
<ID, b> <OPERATOR, =><ID, b><OPERATOR, ^><ID, c>
<PUNCTUATION, ;>
<ID, a> <OPERATOR, +=><ID, b> <PUNCTUATION, ;>
<ID, c> <OPERATOR, /=> <ID, a> <PUNCTUATION, ;>
<KEYWORD, return> <ID, c> <PUNCTUATION, ;>
<SPECIAL SYMBOL, }>

* Every token is a doublet showing the token class and the specific token information
* The output is generated as one token per line. It has been rearranged here for better readability.