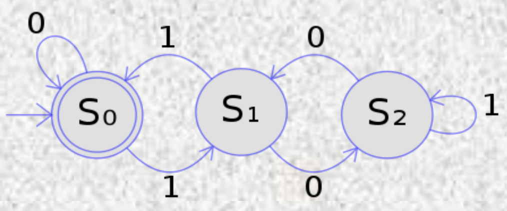
Sheet 2 (Scanner using Regular Expression (RE) and Finite Automaton (FA))

Q1) Given the following ***RE***. Convert each one to its corresponding ***FA*,** then write a C++ code that accepts each one of them.

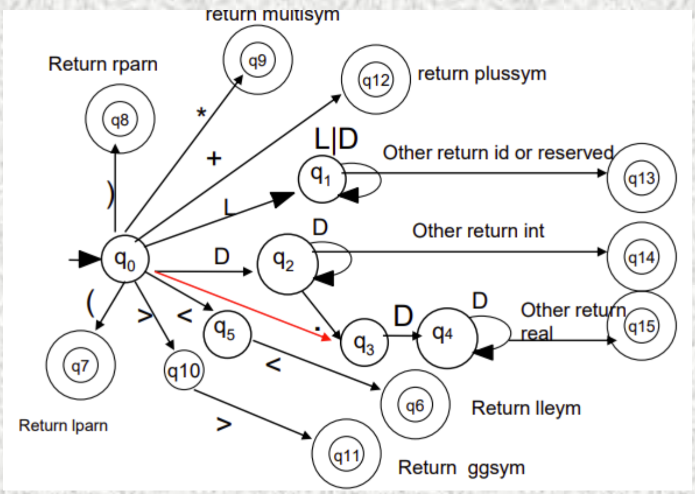
[Don't use file I/O, the input will be assigned in a string variable]

1. $a(\mathrm{b}|\mathrm{a})^*$
2. $(letter \mid \_)\ (letter \mid digit \mid \_)^*$
3. $digit\ digit * .\ digit\ digit^*$
4. $"(L|D|\backslash")^*"$

Q2) Given the following ***FA*** that accepts any binary number divisible by 3**,** write a corresponding C++ code that accepts these binary numbers.



Q3) Given the following ***FA*,** write a corresponding C++ code that simulates this ***FA***.



Q4) Consider the following BNF grammar for the ***Wren*** language. Convert it to its corresponding ***FA,*** then write a C++ code that simulates it for accepting any valid token.

$< program >::= \boldsymbol{program}\ < ident >\ \boldsymbol{is}\ < block >$
$< block >::= < declaration - seq >\ \boldsymbol{begin}\ < command - seq >\ \boldsymbol{end}$
$< declaration - seq >::=\ \varepsilon \mid\ < declaration - seq >< declaration >$
$< declaration >::=\ \boldsymbol{var}\ < var - list >: < type >;$
$< type >::=\ \boldsymbol{integer} \mid \boldsymbol{Boolean}$
$< var - list >::= < var > \mid < var >, < var - list >$
$< command - seq >::= < command > \mid < command - seq >; < command >$
$< command >::= < var >:= < expr > \mid \boldsymbol{read}\ < var > \mid \boldsymbol{write}\ < int - expr > \mid \boldsymbol{if}\ < bool - expr >\ \boldsymbol{then}\ < command - seq >\ \boldsymbol{end\ if} \mid \boldsymbol{if}\ < bool - expr >\ \boldsymbol{then}\ < command - seq >\ \boldsymbol{else}\ < command - seq >\ \boldsymbol{end\ if} \mid \boldsymbol{while}\ < bool - expr >\ \boldsymbol{do}\ < command - seq >\ \boldsymbol{end\ While}$
$< expr >::= < int - expr > \mid < bool - expr >$
$< int - expr >::= < term > \mid < int - expr >< weak\ op >< term >$
$< term >::= < element > \mid < term >< strong\ op >< element >$
$< element >::= < numeral > \mid < var > \mid (< int - expr >) \mid - < element >$
$< bool - expr >::= < bool - term > \mid < bool - expr >\ \boldsymbol{or}\ < bool - term >$

$< bool - term >::= < bool - element > \mid < boo - term >\ \boldsymbol{and}\ < bool - element >$

Dr. Wael Zakaria (Wael.Zakaria@sci.asu.edu.eg)

$< element >::= < var > \mid < comparison > \mid \textbf{not} \; (< bool - expr >) \mid \textbf{true} \mid \textbf{false}$
$< comparison >::= < int - expr > < relation > < int - expr >$
$< var >::= < ident >$
$< relation >::= < \mid <= \mid <> \mid = \mid > \mid >=$
$< weak\ op >::= + \mid -$
$< strong\ op >::= * \mid /$
$< ident >::= < letter > \mid < ident > < letter > \mid < ident > < digit >$
$< letter >::= \textbf{a} \mid \textbf{b} \mid \textbf{c} \mid \textbf{d} \mid \textbf{e} \mid \textbf{f} \mid \textbf{g} \mid \textbf{h} \mid \textbf{i} \mid \textbf{j} \mid \textbf{k} \mid \textbf{l} \mid \textbf{m} \mid \textbf{n} \mid \textbf{o} \mid \textbf{p} \mid \textbf{q} \mid \textbf{r} \mid \textbf{s} \mid \textbf{t} \mid \textbf{u} \mid \textbf{v} \mid \textbf{w} \mid \textbf{x} \mid \textbf{y} \mid \textbf{z}$
$< numeral >::= < digit > \mid < numeral > < digit >$
$< digit >::= \textbf{0} \mid \textbf{1} \mid \textbf{2} \mid \textbf{3} \mid \textbf{4} \mid \textbf{5} \mid \textbf{6} \mid \textbf{7} \mid \textbf{8} \mid \textbf{9}$

Q5) Consider we want to add the following rules to Wren language, modify the the ***FA*** of the ***Wren*** language, then write a C++ code the simulates the new one.

$< Declaration >::= var \; < ident > \; [< numeral > .. < numeral >] : < type >$

$< command >::= < ident > [< ident > \mid < numeral >] := < expr > \mid$

$< ident > := < ident > [< ident > \mid < numeral >] \mid$

$< ident > [< ident > \mid < numeral >] =$

$< ident > [< ident > \mid < numeral >]$