Programming Languages - Homework 1

Write C/C++ functions that builds and runs a FSA and an LR parser, and runs regular expressions.

- Skeleton codes are provided with the assignment.
- **DO NOT change the `fsa_main.cc`, `lr_parser_main.cc`, and `regexp_main.cc` files.**

1.1. Write a C/C++ function that builds a DFA from a finite state automaton definition. [50pts]

- The input file structure is the accept states at the first line, followed by (state, next_state, input_char) triplets. The `epsilon' move is marked as "#" in the text file.

```
3 4
1 3 #
1 2 a
2 2 bc
2 4 b
3 2 #
3 4 a
4 3 ac
```

- Implement the `RunFSA` and `BuildFSA` functions so that it can process both DFA and NFA definitions.

1.2. Write C/C++ functions : a function that loads an LR parsing table to build an LR parser, and a function that runs it on input token strings and returns the acceptance (true/false). [50pts]

- The LR parser table file structure is as follows:

```
num_table_elements  num_rules
state symbol action next_state
...
rule_id lhs_symbol num_rhs
...
```

- Refer description in the header file `lr_parser.h` for more details.
- Design the `LRParser` structure in `lr_parser.h`.
- Implement the `BuildLRParser` function that builds `LRParser` structure using the given table elements.
- Implement the `RunLRParser` function so that it returns the acceptance of the given token string.

1.3. Write a C/C++ function that builds DFA from a regular expression string. [50pts]

- Design and implement the parser that builds NFA from the regular expression, consisting of single characters (`abc`), any character (`.`), set of characters (`[abc]`), OR (`a|b`), zero-or-more repetition ((`a*`), and group (`(abc)`).

```
ab|cd    : ab, cd
a(b|c)d  : abd, acd
a.*b     : ab, acb, axyzb, ...
(a(b.c)*|de)f  : af, def, abxcf, abxcbycf, ...
[abc]*def  : adef, bdef, cdef, aadef, abdef, ...
```

- Convert the built NFA into DFA and match the input string with the regular expression.

Due: **May 31 (Wed) 11:59 pm**

– Zip the source code (ONLY .h, .cc and Makefile; absolutely no executable or object files) and submit it in ezhub (portal).

– The program must run on the Linux server (csedev.hanyang.ac.kr).

```
S => E$
1. E => T
2. E => T|E
3. T => F            Follow Sets
4. T => FT           Follow(E) = $ ) |
5. F => C            Follow(T) = $ ) |
6. F => C*           Follow(F) = $ ) |
7. C => (E)          Follow(C) = $ ) | *
8. C => a            Follow(A) = ]
9. C => [A]
10. A => aA
11. A => a
```

[0] S => .E$ E => .T E => .T|E T => .F T => .FT F => .C F => .C* C => .(E) C => .a C => .[A]	[1] S => E.$	[2] E => T. E => T.|E	[3] T => F. T => F.T T => .F T => .FT F => .C F => .C* C => .(E) C => .a C => .[A]	[4] F => C. F => C.*
	[0]-E	[0]-T	[0]-F	[0]-C
[5] C => (.E) E => .T E => .T|E T => .F T => .FT F => .C F => .C* C => .(E) C => .a C => .[A]	[6] C => a.	[7] C => [.A] A => .aA A => .a	[8] E => T|.E E => .T E => .T|E T => .F T => .FT F => .C F => .C* C => .(E) C => .a C => .[A]	[9] T => FT.
[0]-(	[0]-a	[0]-[	[2]-|	[3]-T
[10] F => C*.	[11] C => (E.)	[12] C => [A.]	[13] A => a.A A => a. A => .a	[14] E => T|E.
[4]-*	[5]-E	[7]-A	[7]-a	[8]-E
[15] C => (E).	[16] C => [A].	[17] A => aA.	[18] A => a.	
[11]-)	[12]-]	[13]-A	[13]-a	

0	0	0	0	0	0	0	1	2	2	3	3	3	3	3	3	3
E	T	F	C	(	a	[	$	<E>	|	<T>	T	F	C	(	a	[
G1	G2	G3	G4	S5	S6	S7	**	R1	S8	R3	G9	G3	G4	S5	S6	S7

4	4	5	5	5	5	5	5	5	6	7	7	8	8	8	8	8
<F>	*	E	T	F	C	(	a	[	<C>	A	a	E	T	F	C	(

R5	S10	G11	G2	G3	G4	S5	S6	S7	R8	G12	S13	G14	G2	G3	G4	S5

8	8	9	10	11	12	13	13	13	14	15	16	17	18			
a	[	<T>	<F>	)	]	<A>	A	a	<E>	<C>	<C>	<A>	<A>			
S6	S7	R4	R6	S15	S16	R11	G17	S18	R2	R7	R9	R10	R11			

Follow Sets
Follow(E) = $) |
Follow(T) = $) |
Follow(F) = $) |
Follow(C) = $) | *
Follow(A) =]

	a	(	)	|	*	[	]	$		E	T	F	C	A
0	S6	S5				S7				1	2	3	4	
1								**						
2			R1	R1				R1						
3	S6	S5	R3	R3		S7		R3			9	3	4	
4			R5	R5	S10			R5						
5	S6	S5				S7				11	2	3	4	
6			R8	R8	R8			R8						
7	S13													12
8	S6	S5				S7				14	2	3	4	
9			R4	R4				R4						
10			R6	R6				R6						
11			S15											
12							S16							
13							R11							17
14			R2	R2				R2						
15			R7	R7	R7			R7						
16			R9	R9	R9			R9						
17							R10							
18							R11							