ASSIGNMENT 3

Part – I

Use flex to redo **Assignment 2.**

Part – II

In this part, you are expected to write the flex code to perform lexical analysis for a language subset of Relational Algebra (RA) Queries consisting of the following operations: {SELECT, PROJECT, CARTESIAN PRODUCT, EQUI-JOIN}. Syntax for these operations are as follows:

I. SELECT

Syntax: SELECT <*condition*> (*Table_Name*)
-*condition* may be simple(eg 1.a) /compound(eg 1.c)

Eg	1.a) SELECT < salary<200 > (Employee)	Output: Valid Syntax
	1.b) SELECT < name='John' AND salary<200 > (Employee)	Output: Valid Syntax
	1.c) SELECT (Employee) < salary<200 >	Output: Invalid Syntax

II. PROJECT

Syntax: PROJECT <*attribute_list*> (*Table_Name*)
-*attribute_list* may be contain single (eg 2.a) /multiple attributes (eg 2.b).

Eg	2.a) PROJECT < salary> (Employee)	Output: Valid Syntax
	2.b) PROJECT < name, salary> (Employee)	Output: Valid Syntax
	2.c) PROJECT (Employee) <salary, name>	Output: Invalid Syntax

III. CARTESIAN PRODUCT

Syntax: < *Table_Name_1* > CARTESIAN_PRODUCT < *Table_Name_2* >

Eg	3.a) (Employee) CARTESIAN_PRODUCT (Department)	Output: Valid Syntax
	3.b) CARTESIAN_PRODUCT (Employee) (Department)	Output: Invalid Syntax
	3.c) (Employee) CARTESIAN_PRODUCT	Output: Invalid Syntax

IV. EQUI-JOIN

Syntax: < *Table_Name_1* > EQUI_JOIN <*condition*> < *Table_Name_2* >

Eg	4.a) (Employee) EQUI_JOIN <Employee.empId = Department.eId>(Department)	Output: Valid Syntax
	4.b) (Employee) EQUI _JOIN (Department)	Output: Invalid Syntax
	4.c) EQUI _JOIN (Employee) <Employee.empId = Department.eId>(Department)	Output: Invalid Syntax

Further, your program should use flex to generate the scanner and bison to generate the parser. The bison parser should generate target code in C language for your input RA program, considering all database tables to be simply represented by text files in comma separate .csv format. The generated C code should be compilable using gcc. Outputs of all queries should be directed to stdout.