Semester Assignment for System Programming (Documentation)

Submitted by: Shubham Jha

Rollno: 2019CSC1051

Files included along the documentation:

1. spassgn.l (Flex file)
2. spassgn.y (Bison file)
3. sqlite3.h
4. sqlite3.c
5. sqlite3.o
6. assignment.db
7. lex.yy.c
8. spassgn.tab.h
9. spassgn.tab.c
10. parser

Files 1 and 2 are the respective lex and yacc files used to code the parser. 3,4 & 5 are the necessary sqlite3 database files used. File 6 is the database used in my code. 7 is the file generated by lex on running file 1. 8 and 9 are the files generated on running file 2. File 10 is our resultant parser that performs all the required tasks as specified by the question.

How to run:

Prerequisites: Flex and Bison should be installed. SQLite files should be present in same folder for convenience in paths. A SQLite3 database should be ready with tables Formulae, FormulaFields and salary with appropriate data.

To get the parser the following commands should be run in the terminal:

```
bison -d spassgn.y

flex spassgn.l

gcc  -o  parser  lex.yy.c  spassgn.tab.c  sqlite3.o

parser    (for windows)
```

Why SQLite3?

I tried to work with mysql. Very few resources available on the internet to link c/c++ with mysql as well as requirement of some specific editors made me look for other choices.

SQLite3 was easy to work with via C and also had good resources available that made this DBMS easy to implement in the code.

Working of the Flex & Bison codes to generate parser

Flex:

The flex file contains 3 sections. In the first section basic C library stdio is included. Also, the tab.h file generated by the bison file is included. In the second section the tokens are defined. They are a variable of type (for eg. <A1>), numbers, parenthesis and the basic arithemetic operators.

In the third section yywrap() function is included. It is called when input is exhausted. It returns 1 if input is done.

Bison:

The bison file also has 3 sections. In the first section basic C libraries are included that are needed in writing C code. Also sqlite3.h header file is included for definition of sqlite3 functions necessary to link DB and C code. It also contains definitions for few functions and a YY Buffer State that is used in the code to give input to yylex.

In the second section, there is our CFG that checks if our grammar is correct or not. If the uppermost production(rootnode->some production) is executed this will confirm that our parse tree is complete and the expression is correct. In any other case yyerror() is called that prints that the expression is incorrect. There are 16 shift/reduce conflicts in the CFG. They are all resolved by giving preference to shift.

The third section of this file is very important and contains all the C code that makes the brain of the parser. Here are few important points about this section:

i. There are four functions: storevariable(), replace(), callback() and main(). Storevariable stores any variable that is returned as token from the lex code and stores it in the odd index of the global array values. Then it retrieves the column name of that variable in salary table and stores it in even index. Replace() function replaces the variables in formula with the columnname using the values array. Callback() is used to print all the values from the sql table.

ii. In the main function, we first establish the db connection. The user inputs a id value and formula for that value is retrived from formulae table.

iii. The output from above is stored in ‘input’ variable and passed to the yyparse(and then to yylex) using yy_scan_string(). After this parse tree is constructed and correctness of expression is checked.

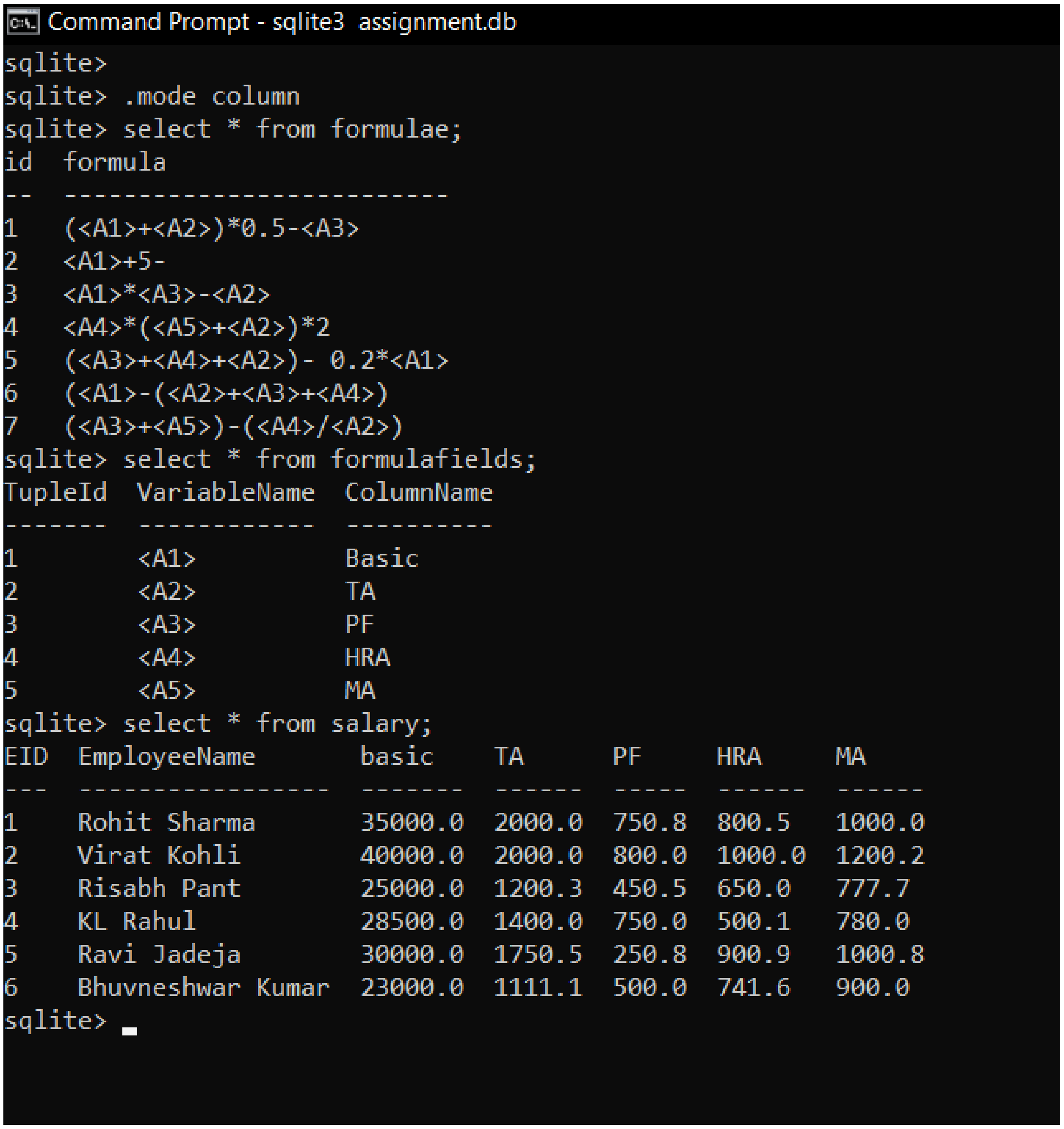
iv. If the expression is correct then, replace function is called and the replaced formula is stored in ‘fr’ variable and eventually concatenated to give out the required sql query.

v. The user inputs range for which the query is to be run in the salary table and then the output is displayed.

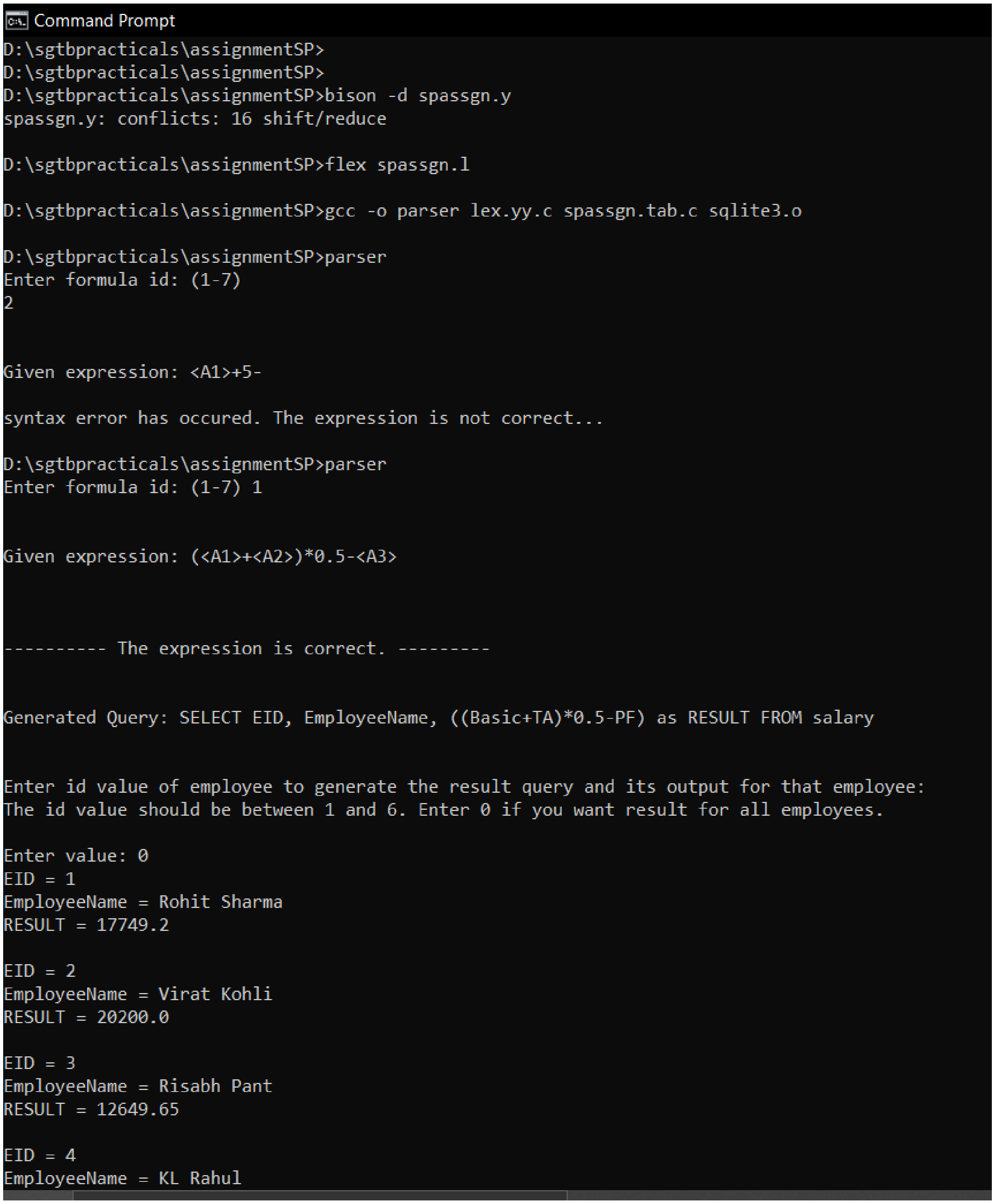
Conclusion:

A parser that takes a formula and replaces it with respective column names and runs the query has been created. Its applications as suggested in the question include: in scripts such as php, as prepared query within a DBMS, in host languages of DBMS such as TCL or C.

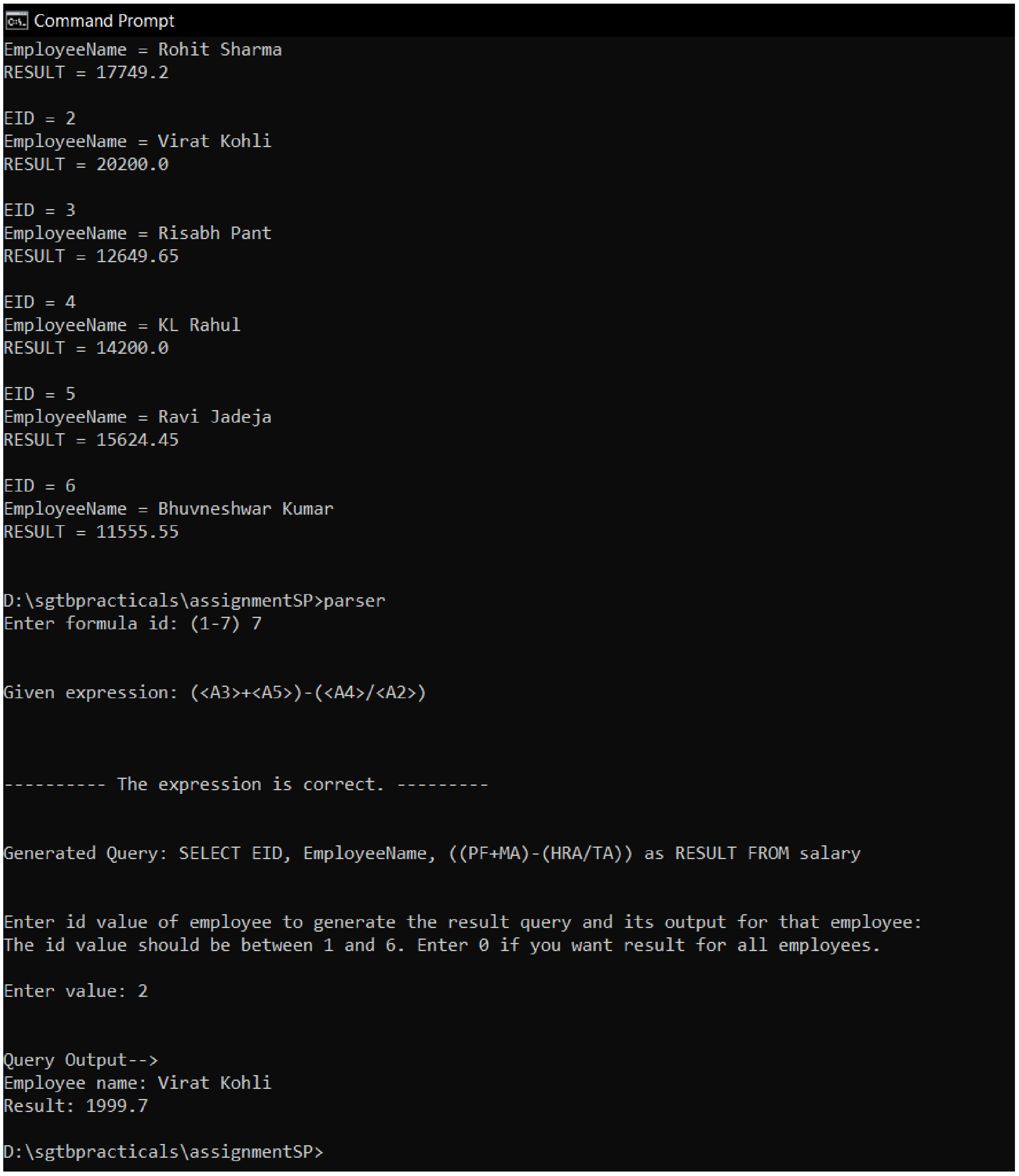
Screenshots of Output:


Command Prompt - sqlite3 assignment.db

```
sqlite>
sqlite> .mode column
sqlite> select * from formulae;
id  formula
--  --------------------------
1   (<A1>+<A2>)*0.5-<A3>
2   <A1>+5-
3   <A1>*<A3>-<A2>
4   <A4>*(<A5>+<A2>)*2
5   (<A3>+<A4>+<A2>)- 0.2*<A1>
6   (<A1>-(<A2>+<A3>+<A4>)
7   (<A3>+<A5>)-(<A4>/<A2>)
sqlite> select * from formulafields;
TupleId  VariableName  ColumnName
-------  ------------  ----------
1        <A1>          Basic
2        <A2>          TA
3        <A3>          PF
4        <A4>          HRA
5        <A5>          MA
sqlite> select * from salary;
EID  EmployeeName       basic    TA      PF     HRA     MA
---  -----------------  -------  ------  -----  ------  ------
1    Rohit Sharma       35000.0  2000.0  750.8  800.5   1000.0
2    Virat Kohli        40000.0  2000.0  800.0  1000.0  1200.2
3    Risabh Pant        25000.0  1200.3  450.5  650.0   777.7
4    KL Rahul           28500.0  1400.0  750.0  500.1   780.0
5    Ravi Jadeja        30000.0  1750.5  250.8  900.9   1000.8
6    Bhuvneshwar Kumar  23000.0  1111.1  500.0  741.6   900.0
sqlite>
```



```
Command Prompt
D:\sgtbpracticals\assignmentSP>
D:\sgtbpracticals\assignmentSP>
D:\sgtbpracticals\assignmentSP>bison -d spassgn.y
spassgn.y: conflicts: 16 shift/reduce

D:\sgtbpracticals\assignmentSP>flex spassgn.l

D:\sgtbpracticals\assignmentSP>gcc -o parser lex.yy.c spassgn.tab.c sqlite3.o

D:\sgtbpracticals\assignmentSP>parser
Enter formula id: (1-7)
2


Given expression: <A1>+5-

syntax error has occured. The expression is not correct...

D:\sgtbpracticals\assignmentSP>parser
Enter formula id: (1-7) 1


Given expression: (<A1>+<A2>)*0.5-<A3>




---------- The expression is correct. ---------


Generated Query: SELECT EID, EmployeeName, ((Basic+TA)*0.5-PF) as RESULT FROM salary


Enter id value of employee to generate the result query and its output for that employee:
The id value should be between 1 and 6. Enter 0 if you want result for all employees.

Enter value: 0
EID = 1
EmployeeName = Rohit Sharma
RESULT = 17749.2

EID = 2
EmployeeName = Virat Kohli
RESULT = 20200.0

EID = 3
EmployeeName = Risabh Pant
RESULT = 12649.65

EID = 4
EmployeeName = KL Rahul
```



```
Command Prompt
EmployeeName = Rohit Sharma
RESULT = 17749.2

EID = 2
EmployeeName = Virat Kohli
RESULT = 20200.0

EID = 3
EmployeeName = Risabh Pant
RESULT = 12649.65

EID = 4
EmployeeName = KL Rahul
RESULT = 14200.0

EID = 5
EmployeeName = Ravi Jadeja
RESULT = 15624.45

EID = 6
EmployeeName = Bhuvneshwar Kumar
RESULT = 11555.55


D:\sgtbpracticals\assignmentSP>parser
Enter formula id: (1-7) 7


Given expression: (<A3>+<A5>)-(<A4>/<A2>)



---------- The expression is correct. ---------


Generated Query: SELECT EID, EmployeeName, ((PF+MA)-(HRA/TA)) as RESULT FROM salary


Enter id value of employee to generate the result query and its output for that employee:
The id value should be between 1 and 6. Enter 0 if you want result for all employees.

Enter value: 2


Query Output-->
Employee name: Virat Kohli
Result: 1999.7

D:\sgtbpracticals\assignmentSP>
```