**Roll No: 1803067**

**Lab Performance Test 2**

**Lab Task Q1a**

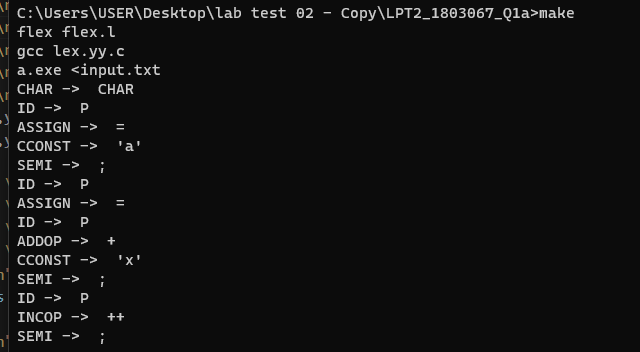
**Question:**

**Q1.** Consider the following code snippet:  **CHAR P = 'a'; P = P + 'x'; P++;**     a) Perform Lexical Analysis on the given code snippet. b) Perform Syntax Analysis on the given code snippet. c) Perform Semantic Analysis on the given code snippet.

**Solution (Bold your own written code):**

|  |
| --- |
| %option noyywrap  %{  // roll : 1803067  %}  alpha     [a-zA-Z\_]  digit     [0-9]  alnum     {alpha}|{digit}  print     [ -~]  ID        {alpha}{alnum}\*  ICONST    [0-9]{digit}\*  FCONST    {digit}\*"."{digit}+  CCONST    (\'{print}\')  STRING    \"{print}\*\"  %%  "//".\*        { }  "INT"       { printf("INT ->  %s \n",yytext); }  "FLAOT"    {printf("DOUBLE ->  %s \n",yytext); }  "CHAR"      {printf("CHAR ->  %s \n",yytext);  }  "++"       {printf("INCOP ->  %s \n",yytext);  }  "+"       {printf("ADDOP ->  %s \n",yytext);  }  "-"       {printf("SUBOP ->  %s \n",yytext); }  "\*"       {printf("MULOP ->  %s \n",yytext);  }  "/"       {printf("DIVOP ->  %s \n",yytext);  }  "=="      {printf("EQUOP ->  %s \n",yytext);  }  ">"       {printf("GT ->  %s \n",yytext); }  "<"       {printf("LT ->  %s \n",yytext); }  "("       {printf("LPAREN ->  %s \n",yytext); }  ")"       {printf("RPAREN ->  %s \n",yytext);  }  "{"       {printf("LBRACE ->  %s \n",yytext); }  "}"       {printf("RBRACE ->  %s \n",yytext);  }  ";"       {printf("SEMI ->  %s \n",yytext); }  "="       { printf("ASSIGN ->  %s \n",yytext);}  {ID}        {printf("ID ->  %s \n",yytext);  } // {strcpy(yylval.str\_val, yytext);  return ID; }  {ICONST}    {printf("ICONST ->  %s \n",yytext); }  {FCONST}    {printf("FCONST ->  %s \n",yytext); }  {CCONST}    { printf("CCONST ->  %s \n",yytext);}  {STRING}    {printf("STRING ->  %s \n",yytext);}  "\n"        {  }  [ \t\r\f]+  .       { printf("Unrecognized character"); }  %%  int main()  {      yylex();      return 0;  } |

**Output (Screen/SnapShot):**

****

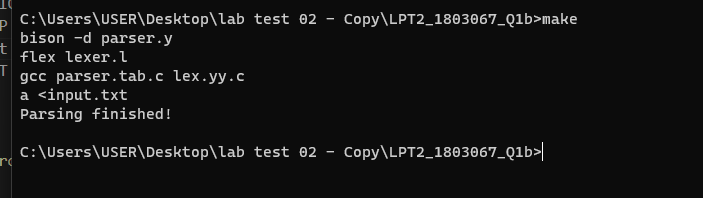
**Lab Task Q1b**

**Question:**

**Solution (Bold your own written code):**

|  |
| --- |
| Lex file :  %option noyywrap  %{      #include <stdio.h>      #include <stdlib.h>      #include <string.h>      #include "parser.tab.h"        int lineno = 1; // initialize to 1      void yyerror();  %}  alpha     [a-zA-Z\_]  digit     [0-9]  alnum     {alpha}|{digit}  print     [ -~]  delim [ \t\n]  ws {delim}+  ID        {alpha}{alnum}\*  ICONST    [0-9]{digit}\*  FCONST    {digit}\*"."{digit}+  CCONST    (\'{print}\')  STRING    \"{print}\*\"  %%  "//".\*        { }  "INT"       { return INT; }  "FLAOT"    { return FLAOT; }  "CHAR"      { return CHAR; }  "++"      { return INCOP; }  "+"       { return ADDOP; }  "-"       { return SUBOP; }  "\*"       { return MULOP; }  "/"       { return DIVOP; }  "=="      { return EQUOP; }  ">"       { return GT; }  "<"       { return LT; }  "%"     {return MOD ;}  ":"     {return COLON;}  "("       { return LPAREN; }  ")"       { return RPAREN; }  "{"       { return LBRACE; }  "}"       { return RBRACE; }  ";"       { return SEMI; }  "="       { return ASSIGN; }  {ID}        {strcpy(yylval.str\_val, yytext);  return ID;  }  {ICONST}    {return ICONST;}  {FCONST}    {return FCONST;}  {CCONST}    {return CCONST;}  "\n"        { lineno += 1; }  [ \t\r\f]+  .       { yyerror("Unrecognized character"); }  %% |
| Bison file :  %{      // roll : 1803067      #include <stdio.h>      #include <stdlib.h>      #include <string.h>      #include"symtab.c"      void yyerror();      extern int lineno;      extern int yylex();      /\*      void insert(char\* name, int type)      int idcheck(char\* name)      int gettype(char \*name)      int typecheck(int type1, int type2)          UNDEF\_TYPE      INT\_TYPE      REAL\_TYPE      CHAR\_TYPE      typename[]        \*/  %}    %union  {      char str\_val[100];      int int\_val;  }    %token INT FLAOT CHAR INCOP  %token ADDOP SUBOP MULOP DIVOP EQUOP LT GT ID  %token LPAREN RPAREN LBRACE RBRACE SEMI ASSIGN FUNCTION RET BEG MOD COLON  %token FCONST END ICONST  %token CCONST  //%type<str\_val> ID  //%type<int\_val>    %left LT GT             /\*LT GT has lowest precedence\*/  %left ADDOP SUBOP  %left MULOP DIVOP  %left MOD           /\*MULOP has highest precedence\*/  %start code  %%  code : code code\_ | ;  code\_ : dec SEMI | assignment SEMI ;  dec : type ID ASSIGN exp;  exp : exp op T | exp INCOP | T  ;  T : ID | ICONST |FCONST | CCONST ;  op : ADDOP | SUBOP | MULOP |DIVOP ;  assignment : ID ASSIGN exp | exp INCOP ;  type : INT | CHAR | FLAOT ;  %%  void yyerror ()  {      printf("Syntax error at line %d\n", lineno);      exit(1);  }  int main (int argc, char \*argv[])  {      yyparse();      printf("Parsing finished!\n");      return 0;  } |

**Output (Screen/SnapShot):**

****

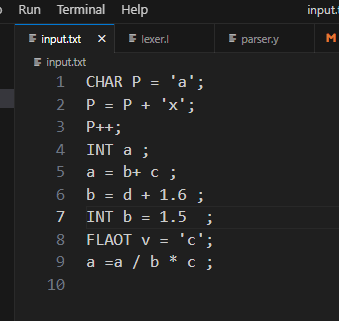
**Lab Task Q1c**

**Question:**

**Solution (Bold your own written code):**

|  |
| --- |
| %option noyywrap  %{      #include <stdio.h>      #include <stdlib.h>      #include <string.h>      #include "parser.tab.h"        int lineno = 1; // initialize to 1      void yyerror();  %}  alpha     [a-zA-Z\_]  digit     [0-9]  alnum     {alpha}|{digit}  print     [ -~]  delim [ \t\n]  ws {delim}+  ID        {alpha}{alnum}\*  ICONST    [0-9]{digit}\*  FCONST    {digit}\*"."{digit}+  CCONST    (\'{print}\')  STRING    \"{print}\*\"  %%  "//".\*        { }  "INT"       { return INT; }  "FLAOT"    { return FLAOT; }  "CHAR"      { return CHAR; }  "++"      { return INCOP; }  "+"       { return ADDOP; }  "-"       { return SUBOP; }  "\*"       { return MULOP; }  "/"       { return DIVOP; }  "=="      { return EQUOP; }  ">"       { return GT; }  "<"       { return LT; }  "%"     {return MOD ;}  ":"     {return COLON;}  "("       { return LPAREN; }  ")"       { return RPAREN; }  "{"       { return LBRACE; }  "}"       { return RBRACE; }  ";"       { return SEMI; }  "="       { return ASSIGN; }  {ID}        {strcpy(yylval.str\_val, yytext);  return ID;  }  {ICONST}    {return ICONST;}  {FCONST}    {return FCONST;}  {CCONST}    {return CCONST;}  "\n"        { lineno += 1; }  [ \t\r\f]+  .       { yyerror("Unrecognized character"); }  %% |
| %{      // roll : 1803067      #include <stdio.h>      #include <stdlib.h>      #include <string.h>      #include"symtab.c"      void yyerror();      extern int lineno;      extern int yylex();      /\*      void insert(char\* name, int type)      int idcheck(char\* name)      int gettype(char \*name)      int typecheck(int type1, int type2)          UNDEF\_TYPE      INT\_TYPE      REAL\_TYPE      CHAR\_TYPE      typename[]        \*/  %}    %union  {      char str\_val[100];      int int\_val;  }    %token INT FLAOT CHAR INCOP  %token ADDOP SUBOP MULOP DIVOP EQUOP LT GT ID  %token LPAREN RPAREN LBRACE RBRACE SEMI ASSIGN FUNCTION RET BEG MOD COLON  %token FCONST END ICONST  %token CCONST  %type<str\_val> ID  %type<int\_val> type  T exp      %left LT GT             /\*LT GT has lowest precedence\*/  %left ADDOP SUBOP  %left MULOP DIVOP  %left MOD           /\*MULOP has highest precedence\*/  %start code  %%  code : code code\_ | ;  code\_ : dec SEMI | assignment SEMI ;  dec : type ID ASSIGN exp { insert($2 , $1 ); typecheck ( gettype($2) ,$1 ) ; }      | type ID { insert($2 , $1 ); } ;  exp : exp op T { $$= typecheck( $1 , $3) ;  }      | exp INCOP { $$ = $1 ;}      | T { $$ = $1 ; } ;  T : ID  { $$ = gettype($1) ;}      | ICONST  { $$ = INT\_TYPE ;}      | FCONST  { $$ = REAL\_TYPE ;}      | CCONST { $$ = CHAR\_TYPE ;};  op : ADDOP | SUBOP | MULOP |DIVOP ;  assignment : ID ASSIGN exp { typecheck( gettype($1) , $3 ) ;  }      | exp INCOP  ;  type : INT { $$ = INT\_TYPE ;}      | CHAR  { $$ = CHAR\_TYPE ;}      | FLAOT { $$ = REAL\_TYPE ;} ;  %%  void yyerror ()  {      printf("Syntax error at line %d\n", lineno);      exit(1);  }  int main (int argc, char \*argv[])  {      yyparse();      printf("Parsing finished!\n");      return 0;  } |

**Input use**

****

**Output (Screen/SnapShot):**

