

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):

```
C:\Users\USER\Desktop\lab test 02 - Copy\LPT2_1803067_Q1a>make
flex flex.l
gcc lex.yy.c
a.exe <input.txt
CHAR -> CHAR
ID -> P
ASSIGN -> =
CCONST -> 'a'
SEMI -> ;
ID -> P
ASSIGN -> =
ID -> P
ADDOP -> +
CCONST -> 'x'
SEMI -> ;
ID -> P
INCOP -> ++
SEMI -> ;
```

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(yyval.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):

```
C:\Users\USER\Desktop\lab test 02 - Copy\LPT2_1803067_Q1b>make  
bison -d parser.y  
flex lexer.l  
gcc parser.tab.c lex.yy.c  
a <input.txt  
Parsing finished!  
  
C:\Users\USER\Desktop\lab test 02 - Copy\LPT2_1803067_Q1b>|
```

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(yy1val.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 ()

```

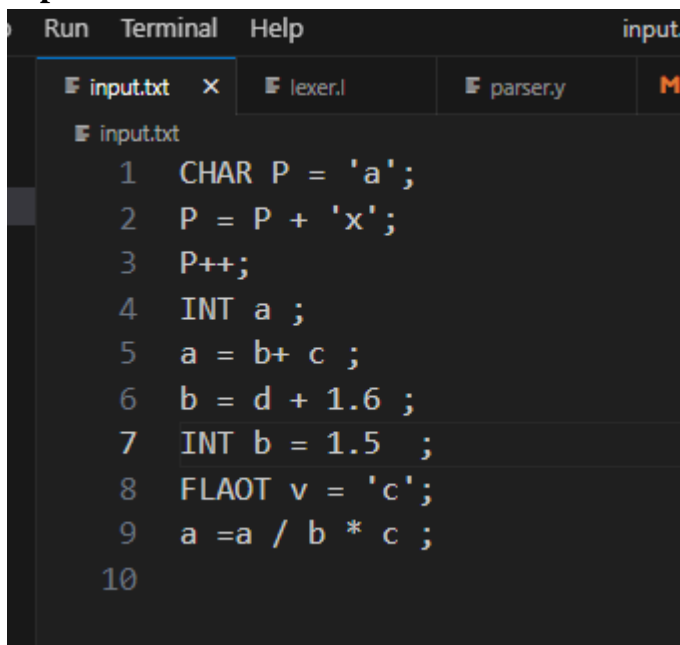
```

{
    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



The screenshot shows a code editor with three tabs: 'input.txt', 'lexer.l', and 'parser.y'. The 'input.txt' tab is active, displaying the following code:

```

1  CHAR P = 'a';
2  P = P + 'x';
3  P++;
4  INT a ;
5  a = b+ c ;
6  b = d + 1.6 ;
7  INT b = 1.5  ;
8  FLAOT v = 'c';
9  a =a / b * c ;
10

```

Output (Screen/SnapShot):

```
C:\Users\USER\Desktop\lab test 02 - Copy\LPT2_1803067_Q1c>make
bison -d parser.y
flex lexer.l
gcc parser.tab.c lex.yy.c
a <input.txt
In line no :1, Inserting "P" with type "CHAR_TYPE" in symbol table.
In line no :4, Inserting "a" with type "INT_TYPE" in symbol table.
In line no :5, ID "b" is not declared.
In line no :5, ID "c" is not declared.
In line no :5, Data type "UNDEF_TYPE" is not matched with Data type "UNDEF_TYPE".
In line no :5, Data type "INT_TYPE" is not matched with Data type "UNDEF_TYPE".
In line no :6, ID "d" is not declared.
In line no :6, Data type "UNDEF_TYPE" is not matched with Data type "REAL_TYPE".
In line no :6, ID "b" is not declared.
In line no :6, Data type "UNDEF_TYPE" is not matched with Data type "UNDEF_TYPE".
In line no :7, Inserting "b" with type "INT_TYPE" in symbol table.
In line no :8, Inserting "v" with type "REAL_TYPE" in symbol table.
In line no :9, ID "c" is not declared.
In line no :9, Data type "INT_TYPE" is not matched with Data type "UNDEF_TYPE".
In line no :9, Data type "INT_TYPE" is not matched with Data type "UNDEF_TYPE".
Parsing finished!
```

```
C:\Users\USER\Desktop\lab test 02 - Copy\LPT2_1803067_Q1c>|
```