Roll No: 1903001

Lab Performance Evaluation [01]

Lab Task Q1

Question:

```c
#include<math.h>

int start()

{

float a=1; float b=2; float c=a+b;

return 0;

}
```

Solution (Bold your own written code):

Makefile

```
main:
    gcc -c -o p1.o p1.c
```

Output (Screen/SnapShot):

```
E:\My_Programs\CSE4102\LPE1\LPE1_1903001_Q1>make
gcc -c -o p1.o p1.c
```

Lab Task Q2a

Question:

ret type int main()
{
var type float a = 9;
var type int b = 5;
if(a+b>10){
var type double c = 90.7;
}
return 0;
}

Solution (Bold your own written code):

FLEX

```
%option noyywrap

%{
    #include <stdio.h>
    #include<stdlib.h>
%}

letter [a-zA-Z]
digit [0-9]
ID (_|{letter})(_|{letter}|{digit})*
ws [ \n]

%%

({ws}) {}

"float" { printf("%s -> Float_Type\n", yytext); }
"int" { printf("%s -> INT_TYPE\n", yytext); }
"double" { printf("%s -> DOUBLE_TYPE\n", yytext); }
"if" { printf("%s -> IF\n", yytext); }
"ret" { printf("%s -> RET\n", yytext); }
"type" { printf("%s -> TYPE\n", yytext); }
"var" { printf("%s -> VAR\n", yytext); }
"return" { printf("%s -> RETURN\n", yytext); }
{digit}*"."{digit}+ { printf("%s -> REAL_NUM\n", yytext); }
{digit}+ { printf("%s -> INT_NUM\n", yytext); }
{ID} { printf("%s -> ID\n", yytext); }
"=" { printf("%s -> ASSIGN\n", yytext); }
">" { printf("%s -> GT\n", yytext); }
"(" { printf("%s -> LP\n", yytext); }
```

```
")" { printf("%s -> RP\n", yytext); }
"+" { printf("%s -> PLUS\n", yytext); }
";" { printf("%s -> SEMI\n", yytext); }
"{" { printf("%s -> LB\n", yytext); }
"}" { printf("%s -> RB\n", yytext); }
%%

int main(){
    yylex();
    return 0;
}
```

Output (Screen/SnapShot):

```
E:\My_Programs\CSE4102\LPE1\LPE1_1903001_Q2a>make
flex lex.l
gcc lex.yy.c
a < in.txt
ret -> RET
type -> TYPE
int -> INT_TYPE
main -> ID
( -> LP
) -> RP
{ -> LB
var -> VAR
type -> TYPE
float -> Float_Type
a -> ID
= -> ASSIGN
9 -> INT_NUM
; -> SEMI
var -> VAR
type -> TYPE
int -> INT_TYPE
b -> ID
= -> ASSIGN
5 -> INT_NUM
; -> SEMI
if -> IF
( -> LP
a -> ID
+ -> PLUS
b -> ID
> -> GT
10 -> INT_NUM
) -> RP
{ -> LB
var -> VAR
type -> TYPE
double -> DOUBLE_TYPE
c -> ID
= -> ASSIGN
90.7 -> REAL_NUM
; -> SEMI
} -> RB
return -> RETURN
0 -> INT_NUM
; -> SEMI
} -> RB
```

Lab Task Q2b

Question:

ret type int main()
{
var type float a = 9;
var type int b = 5;
if(a+b>10){
var type double c = 90.7;
}
return 0;
}

Solution (Bold your own written code):

FLEX

```
%option noyywrap

%{
    // Roll - 1903001
    #include <stdio.h>
    #include<stdlib.h>
    #include "bis.tab.h"
    int lineno=1;
    void yyerror();
%}

letter [a-zA-Z]
digit [0-9]
ID ({letter})({letter}|{digit})*
quo ["]
ws [ ]
sc [ :=+-_]
literal ({quo})({letter}|{digit}|{sc})*({quo})

%%
({ws}) {}

"float" { return(FLOAT); }
"int" { return(INT); }
"double" { return(DOUBLE); }
"if" { return(IF); }
"ret" { return(RET); }
"type" { return(TYPE); }
```

```
"var" { return(VAR); }
"return" { return(RETURN); }
{digit}*"."{digit}+ { return(REAL_NUM); }
{digit}+ { return(INT_NUM); }
{ID} { return(ID); }
"=" { return(ASSIGN); }
">" { return(GT); }
"(" { return(LP); }
")" { return(RP); }
"+" { return(PLUS); }
";" { return(SEMI); }
"{" { return(LB); }
"}" { return(RB); }
%%

// int main(){
//  yylex();
//  return 0;
// }
```

BISON

```
%{
    #include <stdio.h>
    #include <stdlib.h>
    void yyerror();
    extern int lineno;
    extern int yylex();
%}

%union
{
    char str_val[100];
    int int_val;
}

%token FLOAT INT DOUBLE IF RET TYPE VAR RETURN REAL_NUM
%token INT_NUM ASSIGN GT LP RP PLUS SEMI LB RB
%token ID


%start code
%%
code: main_func;
main_func: RET TYPE INT ID LP RP LB statements RB;
statements: statements statement | ;
```

```
statement: declaration
        | conditional
        | return_statement ;
declaration: VAR TYPE dtype ID ASSIGN exp SEMI;
dtype: INT | FLOAT | DOUBLE ;
constant: REAL_NUM | INT_NUM;
exp: ID
    | constant
    | exp PLUS exp
    | exp GT exp ;

conditional: IF LP exp RP LB statements RB ;
return_statement: RETURN constant SEMI ;

%%

int main()
{
    yyparse();
    printf("Parsing Finshed\n");
    return 0;
}

void yyerror(char *err){
    printf("Syntax error at line %d\n", lineno);
    exit(1);
}
```

Output (Screen/SnapShot):

```
E:\My_Programs\CSE4102\LPE1\LPE1_1903001_Q2b>make
bison -d bis.y
bis.y: conflicts: 4 shift/reduce
flex lex.l
gcc bis.tab.c lex.yy.c
a < in.txt



Parsing Finshed
```

Lab Task Q2c

Question:

ret type int main()
{
var type float a = 9;
var type int b = 5;
if(a+b>10){
var type double c = 90.7;
}
return 0;
}

Solution (Bold your own written code):

FLEX

```
%option noyywrap

%{
    // Roll - 1903001
    #include <stdio.h>
    #include<stdlib.h>
    #include "bis.tab.h"
    #include <string.h>
    int lineno = 1;
    void yyerror();
%}

letter [a-zA-Z]
digit [0-9]
ID ({letter})({letter}|{digit})*
quo ["]
ws [ ]
sc [ :=+-_]
literal ({quo})({letter}|{digit}|{sc})*({quo})

%%
({ws}) {}

"float" { return(FLOAT); }
"int" { return(INT); }
"double" { return(DOUBLE); }
"if" { return(IF); }
"ret" { return(RET); }
```

```
"type" { return(TYPE); }
"var" { return(VAR); }
"return" { return(RETURN); }
{digit}*"."{digit}+ { return(REAL_NUM); }
{digit}+ { return(INT_NUM); }
{ID} { strcpy(yylval.str_val, yytext);
    return(ID); }
"=" { return(ASSIGN); }
">" { return(GT); }
"(" { return(LP); }
")" { return(RP); }
"+" { return(PLUS); }
";" { return(SEMI); }
"{" { return(LB); }
"}" { return(RB); }
"\n" {lineno+=1;}
%%

// int main(){
//  yylex();
//  return 0;
// }
```

BISON

```
%{
    // Roll - 1903001
    #include <stdio.h>
    #include <stdlib.h>
    #include <string.h>
    #include "symtab.c"
    void yyerror();
    extern int lineno;
    extern int yylex();
%}

%union
{
    char str_val[100];
    int int_val;
}

%token FLOAT INT DOUBLE IF RET TYPE VAR RETURN REAL_NUM
%token INT_NUM ASSIGN GT LP RP PLUS SEMI LB RB
%token<str_val> ID
```

```
%left GT
%left PLUS


%type<int_val> declaration dtype exp constant

%start code
%%
code: main_func;
main_func: RET TYPE INT ID LP RP LB statements RB;
statements: statements statement | ;
statement: declaration
        | conditional
        | return_statement ;
declaration: VAR TYPE dtype ID ASSIGN exp SEMI
           {
               insert($4, $3);
               typecheck(gettype($4), $6);
           };
dtype: INT {$$=INT_TYPE;}
    | FLOAT {$$=REAL_TYPE;}
    | DOUBLE {$$=REAL_TYPE;}
    ;
constant: REAL_NUM {$$=REAL_TYPE;}
    | INT_NUM {$$=INT_TYPE;}
    ;
exp: ID
    {
        if(idcheck($1))
        {
            $$ = gettype($1);
        }
    }
    | constant {$$=$1;}
    | exp PLUS exp
    {
        $$ = typecheck($1, $3);
    }
    | exp GT exp
    {
        $$ = typecheck($1, $3);
    }
    ;

conditional: IF LP exp RP LB statements RB ;
return_statement: RETURN constant SEMI ;
%%
```

```c
int main()
{
    yyparse();
    printf("Parsing Finshed\n");
    return 0;
}

void yyerror(char *err){
    printf("Syntax error at line %d\n", lineno);
    exit(1);
}
```

Output (Screen/SnapShot):

```
E:\My_Programs\CSE4102\LPE1\LPE1_1903001_Q2c>make
bison -d bis.y
flex lex.l
gcc bis.tab.c lex.yy.c
a < in.txt
In line no 5, Inserting a with type REAL_TYPE in symbol table.
In line no 5, Data type REAL_TYPE is not matched with Data type INT_TYPE.
In line no 7, Inserting b with type INT_TYPE in symbol table.
In line no 9, Data type REAL_TYPE is not matched with Data type INT_TYPE.
In line no 9, Data type UNDEF_TYPE is not matched with Data type INT_TYPE.
In line no 11, Inserting c with type REAL_TYPE in symbol table.
Parsing Finshed

E:\My_Programs\CSE4102\LPE1\LPE1_1903001_Q2c>
```