**Roll No: 1903001**

**Lab Performance Evaluation [01]**

**Lab Task Q1**

**Question:**

#include<math.h>

int start()

{

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

return 0;

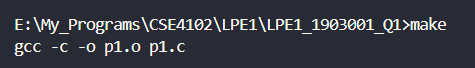
}

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

**Makefile**

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

**Output (Screen/SnapShot):**



**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;

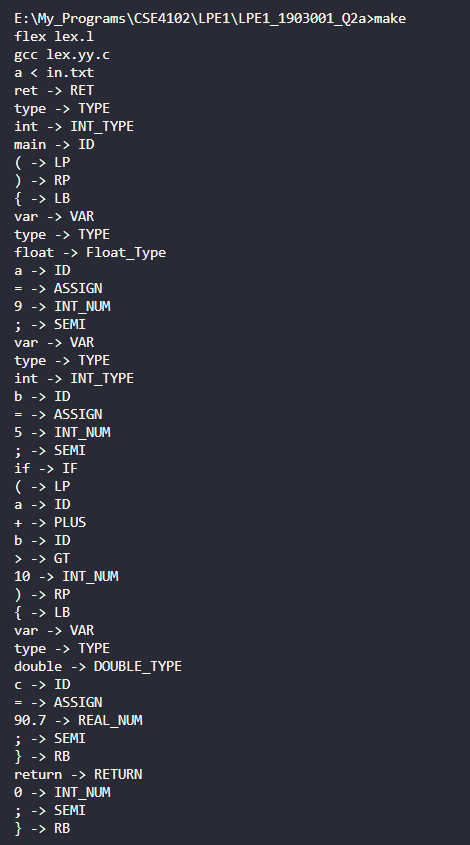
}

**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):**

****

**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):**

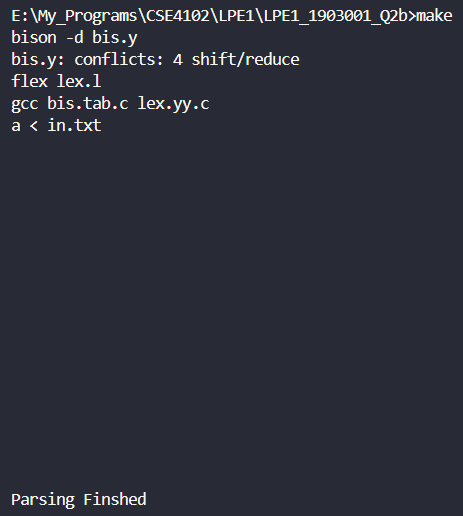
**FLEX**

|  |
| --- |
| %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):**

****

**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 ;  %%  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):**

