GROUP 5

ASSIGNMENT 3

PART 1

lexer.lex

```
%{
#include<bits/stdc++.h>
using namespace std;
set<string> className;
string SPACE = " \n\r\t\f\v";
set<int> obj;
set<int> classDec;
set<int> constDec;
set<int> inherClass;
set<int> opOverload;
string removespace(string s)
{
   int start = s.find_first_not_of(SPACE);
   if (start==-1)
      return s.substr(0 , 0);

   s = s.substr(start);
   int end = s.find_last_not_of(SPACE);
   s = s.substr(0 , end + 1);
   return s;
}
void addClassName(string s){
   s = s.substr(5, s.length());
   s = removespace(s);
   string ans;
```

```
    int i = 0;
    while(isalnum(s[i]) || s[i]=='_'){
        ans+=s[i];
        i++;
    }
    className.insert(ans);

    return;
}
int identifyConstObj(string s){
    string ans;
    int i = 0;
    while(isalnum(s[i]) || s[i]=='_'){
        ans+=s[i];
        i++;
    }
    if(className.find(ans)!=className.end()){
        return 1;
    }
    return 0;
}
%}
%option yylineno
%x CLASS
%x NAMECLASS
%%
```

```
"//".*                                                    { }
[/][*][^*]*[*]+([^*/][^*]*[*]+)*[/]                      { }
["][^"]*["]                                              { }
"class"[ \n]+[a-zA-Z0-9_]+[ \n]+"{"
{addClassName(yytext);classDec.insert(yylineno);}
"class"[ \n]+[a-zA-Z0-9_]+[ \n]*":"[ \n]*[a-zA-Z0-9_ \n,]+"{"
 {addClassName(yytext);classDec.insert(yylineno);inherClass.insert(yylineno);}

operator                                                 {opOverload.insert(yylineno);}
[~][a-zA-Z0-9_]+[ \n]*"("                                { }
[a-zA-Z0-9_]+[ \n]*"("                    { if(identifyConstObj(yytext))constDec.insert(yylineno);}
[a-zA-Z0-9_]+[ \n*]+[a-zA-Z0-9_]+[ \n]*[,;=(]+ { if(identifyConstObj(yytext))obj.insert(yylineno);}
.                                                        {  }
\n                                                       { }
%%

int yywrap(){}
int main(){
yylex();

cout<<"Object Declaration - "<<obj.size()<<endl;
cout<<"Class Definition - "<<classDec.size()<<endl;
cout<<"Constructor Definition - "<<constDec.size()<<endl;
cout<<"Inherited Class Definition - "<<inherClass.size()<<endl;
cout<<"Operator Overloaded function Definition - "<<opOverload.size()<<endl;
return 0;
}
```

PART 2

lexer.lex

```
%option noyywrap
%{
   #include <stdio.h>
   #include "parser.tab.h"
%}

%s HASH

%%
"SELECT"                                {yylval.stringVal = strdup(yytext);return SELECT;}
"<"                                     {yylval.stringVal = strdup(yytext);return LT;}
">"                                     {yylval.stringVal = strdup(yytext);return GT;}
"("                                     {yylval.stringVal = strdup(yytext);return OB;}
")"                                     {yylval.stringVal = strdup(yytext);return CB;}
"PROJECT"                               {yylval.stringVal = strdup(yytext);return PROJECT;}
"CARTESIAN_PRODUCT"                     {yylval.stringVal = strdup(yytext);return CARTESIAN;}
"EQUI_JOIN"                             {yylval.stringVal = strdup(yytext);return EQUI;}
"AND"                                   {yylval.stringVal = strdup(yytext);return AND;}
"OR"                                    {yylval.stringVal = strdup(yytext);return OR;}
[a-zA-Z_][a-zA-Z0-9_]*                  {yylval.stringVal = strdup(yytext);return NAME;}
[0-9]+                                  {yylval.stringVal = strdup(yytext);return DIGIT;}
"'"[a-zA-Z0-9_ ]*"'"                    {yylval.stringVal = strdup(yytext);return STRING;}
"="                                     {yylval.stringVal = strdup(yytext);return EQ;}
"<="                                    {yylval.stringVal = strdup(yytext);return LE;}
">="                                    {yylval.stringVal = strdup(yytext);return GE;}
","                                     {yylval.stringVal = strdup(yytext);return COMMA;}
"."                                     {yylval.stringVal = strdup(yytext);return DOT;}
```

```
[ \n\t]+                                                {}
";"                                                     {yylval.stringVal = strdup(yytext);return SEMI;}
.                                                       {printf("error\n");}
"!="                                                    {yylval.stringVal = strdup(yytext);return NEQ;}

%%
```

parser.y

```
%{
#include <stdio.h>
void yyerror(char *s){
   printf ("Invalid Syntax\n");
}
%}
%union{ char * stringVal; }
%token <stringVal> SELECT LT GT LE GE EQ OB CB PROJECT CARTESIAN NAME EQUI AND OR COMMA
DOT SEMI NEQ STRING DIGIT
%start STATEMENTS
%%
STATEMENTS :        STATEMENT STATEMENTS   {}
            |  STATEMENT          {}

STATEMENT :         SELECT LT CONDITIONS GT OB NAME CB SEMI {printf("Valid Syntax\n");}
            |  PROJECT LT ATTR_LIST GT OB NAME CB SEMI {printf("Valid Syntax\n");}
            |  OB NAME CB CARTESIAN OB NAME CB SEMI    {printf("Valid Syntax\n");}
            |  OB NAME CB EQUI LT EQUI_CONDITION GT OB NAME CB SEMI    {printf("Valid Syntax\n");}
            |  error SEMI { yyerrok; }
```

```
CONDITIONS :        CONDITION AND CONDITIONS      {}
            |  CONDITION OR CONDITIONS       {}
            |  CONDITION                    {}

CONDITION  :        NAME EQ STRING          {}
            |   NAME NEQ STRING          {}
            |  NAME EQ NAME          {}
            |   NAME NEQ NAME          {}
            |  NAME LT NAME          {}
            |  NAME GT NAME          {}
            |  NAME LE NAME          {}
            |  NAME GE NAME          {}
            |  NAME LT DIGIT          {}
            |  NAME GT DIGIT          {}
            |  NAME LE DIGIT          {}
            |  NAME GE DIGIT          {}
            |  NAME EQ DIGIT          {}
            |   NAME NEQ DIGIT          {}

ATTR_LIST  :        NAME COMMA ATTR_LIST          {}
            |  NAME                    {}

EQUI_CONDITION :      NAME DOT NAME EQ NAME DOT NAME  {}
            |   NAME DOT NAME NEQ NAME DOT NAME  {}

%%
```

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

HOW TO RUN

Part 1

lex lexer.lex

g++ lex.yy.c

./a.out < input.cpp

Part 2

bison -d -v parser.y

flex lexer.lex

gcc -w parser.tab.c lex.yy.c -o main

./main < input.in