Name: Mihir Thakkar

Class: TY A

Roll No: 59

Srn: 201901267

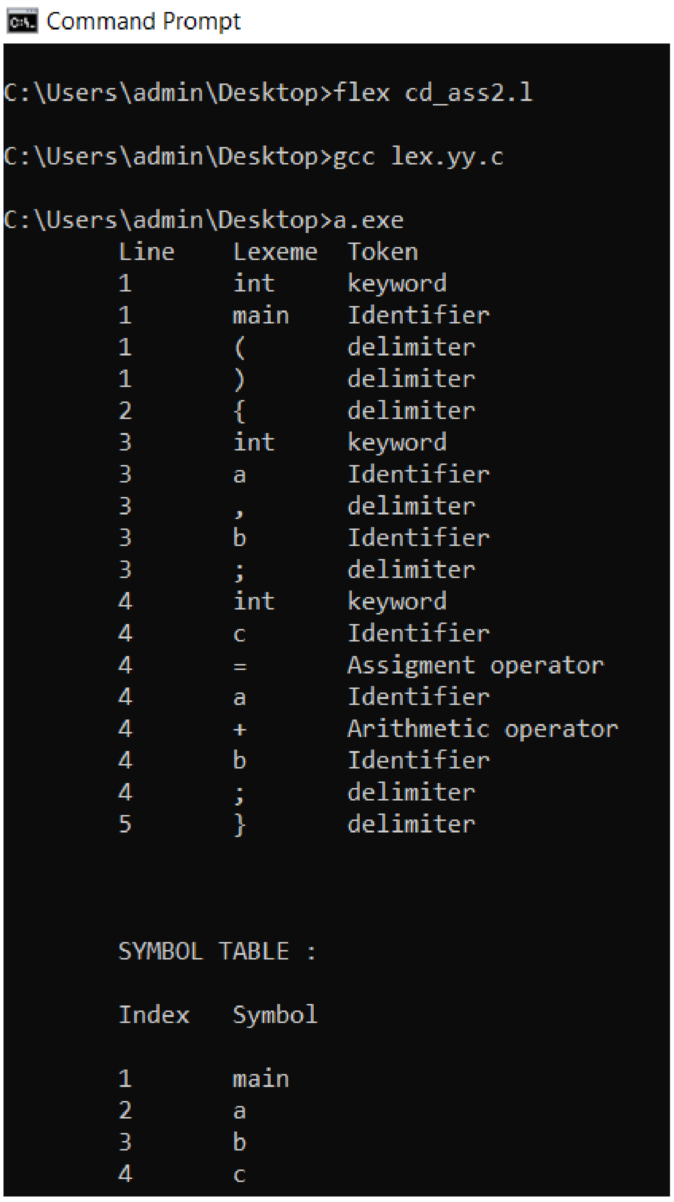
Problem Statement:

Design a Lexical analyzer for the subset of Java Language. Read input from the file. Also create symbol table. Detect any one lexical error. Output in 4 columns Line No, Lexeme, Token and Token Value. Upload single file containing input, output and source code.

Input Code: (C code)

```
int main ( )
{
 int a , b ;
 int c = a + b ;
}
```

Output:



Command Prompt

```
C:\Users\admin\Desktop>flex cd_ass2.l

C:\Users\admin\Desktop>gcc lex.yy.c

C:\Users\admin\Desktop>a.exe
        Line    Lexeme  Token
        1       int     keyword
        1       main    Identifier
        1       (       delimiter
        1       )       delimiter
        2       {       delimiter
        3       int     keyword
        3       a       Identifier
        3       ,       delimiter
        3       b       Identifier
        3       ;       delimiter
        4       int     keyword
        4       c       Identifier
        4       =       Assigment operator
        4       a       Identifier
        4       +       Arithmetic operator
        4       b       Identifier
        4       ;       delimiter
        5       }       delimiter



        SYMBOL TABLE :

        Index   Symbol

        1       main
        2       a
        3       b
        4       c
```

Source Code: (lex code)

```
%{

#include  <stdio.h>

#include <string.h>

int LN=1;

int count=0;

char symbols[100][20];

int symbolPos [100];

char temp[20];

%}

digit [0-9]

letter [A-Za-z_]

%%

"while"|"if"|"else
if"|"else"|"for"|"case"|"return"|"int"|"char"|"float"|"double"|"do"|"void"|"break"|"long"
{fprintf(yyout,"\t%d\t%s\tkeyword\n",LN,yytext);}

{letter}({letter}|{digit})* {fprintf(yyout,"\t%d\t%s\tIdentifier\n",LN,yytext);

strcpy(symbols[count],yytext);

count++;

}

"{"|"}"|"("|")"|";"|"," {fprintf(yyout,"\t%d\t%s\tdelimiter\n",LN,yytext);}

{digit}+ {fprintf(yyout,"\t%d\t%s\tInteger\n",LN,yytext);}

{digit}+\.{digit}+ {fprintf(yyout,"\t%d\t%s\tDecimal\n",LN,yytext);}

"=" {fprintf(yyout,"\t%d\t%s\tAssigment operator\n",LN,yytext);}

"&&"|"||" {fprintf(yyout,"\t%d\t%s\tLogical operator\n",LN,yytext);}

"=="|"<="|">="|"!="|"<"|">" {fprintf(yyout,"\t%d\t%s\tRelational operator\n",LN,yytext);}

"+"|"-"|"*"|"/"|"++"|"--" {fprintf(yyout,"\t%d\t%s\tArithmetic operator\n",LN,yytext);}

"\n" {LN++;}

%%

int yywrap()

{
```

```
return 1;
}


char remove_dups(int count, char array[100][20])
{
   int k, r, h, i, printCount = count;
   char ob[100][20];
strcpy(ob[0],array[0]); h= 1;
for(r= 0 ; r<count ; r++)
{
   k= 0;
   while (k< h)
   {
      if (strcmp(array[r], ob[k]) == 0){
               printCount--;
          break;
          }
       k++;
   }
   if (k==h) {
      strcpy(ob[h],array[r]);
      h++;
   }
}
for(i = 0;i<printCount+1;i++)
printf("\t%d\t%s\n",(i+1),ob[i]);
}
```

```
int main(int argc,char* argv[])

{

int i,j;

char* str[20];

char final[20];

yyin = fopen("cd_ass2.c", "r");

/*yyout = fopen("output.txt","w");*/

///symout = fopen("symbol.txt","w");

printf("\tLine\tLexeme\tToken\n");

yylex();

//printf("%d\n",count);

for(j = 0;j<count;j++)

str[count] = symbols[count];

printf("\n\n\n\tSYMBOL TABLE : \n\n");

printf("\tIndex\tSymbol\n\n");

remove_dups(count,symbols);

/*fclose(yyout);*/

fclose(yyin);

}
```