**CSA1465 – COMPILER DESIGN DAY 2 LAB PROGRAMS**

**QUE 1**

**CODE:**

%{

#include <stdio.h>

int counter = 0;

%}

%%

[a-zA-Z]+ {

if (yyleng > counter) {

counter = yyleng;

}

}

%%

int yywrap() {

return 1; // Indicate that there are no more input files

}

int main() {

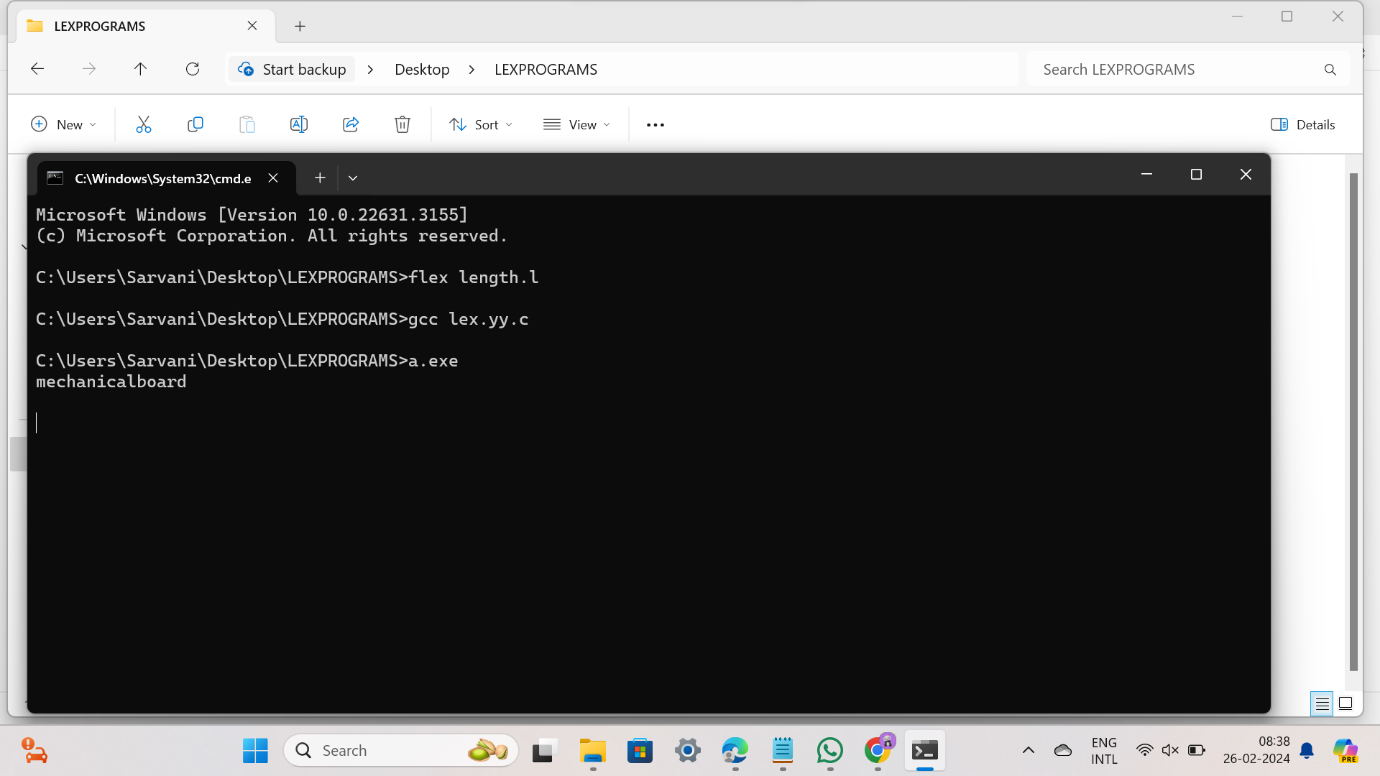
yylex();

printf("largest: %d\n", counter);

return 0;

}

**OUTPUT:**



**QUE 2**

**CODE:**

%%

((http)|(ftp))s?:\/\/[a-zA-Z0-9](.[a-z])+(.[a-zA-Z0-9+=?]) {printf("\nURL Valid\n");}

.+ {printf("\nURL Invalid\n");}

%%

void main()

{

printf("\nEnter URL : ");

yylex();

printf("\n");

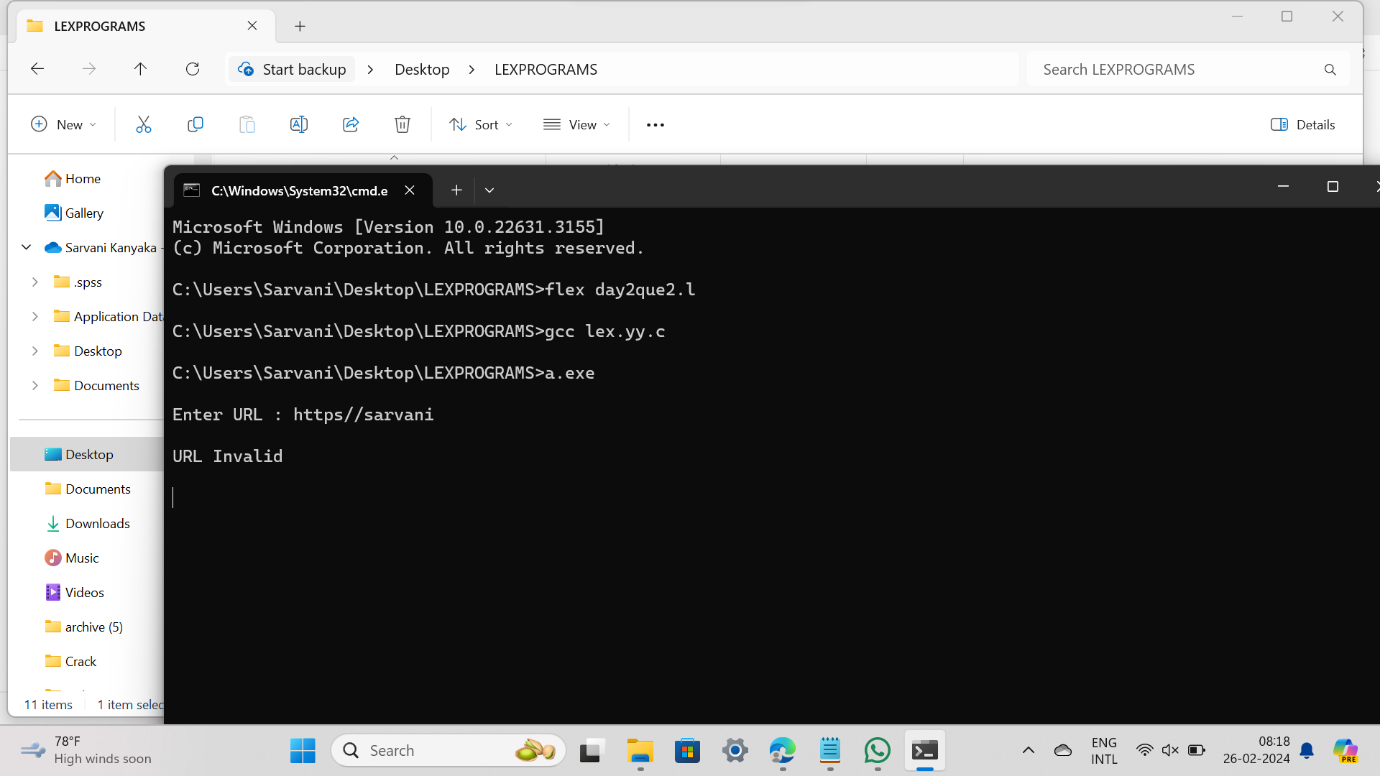
}

int yywrap()

{

}

**OUTPUT:**



**QUE 3**

**CODE:**

%%

((0[1-9])|([1-2][0-9])|(3[0-1]))\/((0[1-9])|(1[0-2]))\/(19[0-9]{2}|2[0-9]{3}) printf("Valid DoB");

.\* printf("Invalid DoB");

%%

int main()

{

yylex();

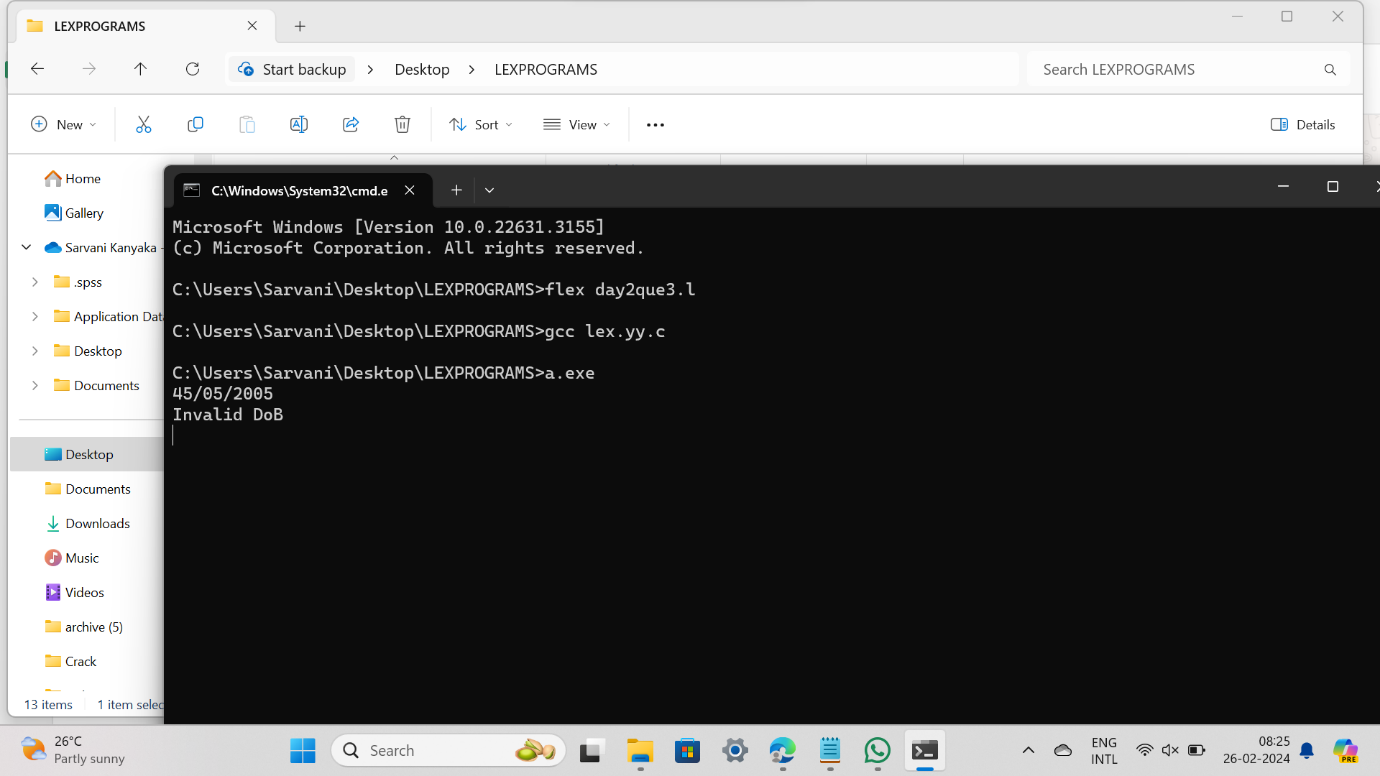
return 0;

}

int yywrap()

{}

**OUTPUT:**



**QUE 4**

**CODE:**

%{

#include <stdio.h>

%}

%%

[a-zA-Z][a-zA-Z0-9]\* { printf("Word: %s\n", yytext); }

"=="|"<"|">"|"<="|">="|"!=" { printf("Relational Operator: %s\n", yytext); }

. ; // Ignore any other characters

%%

int main() {

yylex();

return 0;

}

**QUE 5**

**CODE:**

%{

#include <stdio.h>

#include <string.h>

#define MAX\_BUFFER\_SIZE 1024

%}

%%

"old\_word" { fprintf(yyout, "new\_word"); }

.|\n { fprintf(yyout, "%s", yytext); }

%%

int main(int argc, char \*argv[]) {

if (argc != 4) {

fprintf(stderr, "Usage: %s input\_file output\_file\n", argv[0]);

return 1;

}

FILE \*inputFile = fopen(argv[2], "r");

FILE \*outputFile = fopen(argv[3], "w");

if (!inputFile || !outputFile) {

perror("Error opening files");

return 1;

}

yyin = inputFile;

yyout = outputFile;

yylex();

fclose(inputFile);

fclose(outputFile);

return 0;

}

**QUE 6**

**CODE:**

%{

#undef yywrap

#define yywrap() 1

int f1=0,f2=0;

char oper;

float op1=0,op2=0,ans=0;

void eval();

%}

DIGIT [0-9]

NUM {DIGIT}+(\.{DIGIT}+)?

OP [\*/+-]

%%

{NUM} {

if(f1==0)

{

op1=atof(yytext);

f1=1;

}

else if(f2==-1)

{

op2=atof(yytext);

f2=1;

}

if((f1==1) && (f2==1))

{

eval();

f1=0;

f2=0;

}

}

{OP} {

oper=(char) \*yytext;

f2=-1;

}

[\n] {

if(f1==1 && f2==1)

{

eval;

f1=0;

f2=0;

}

}

%%

int main()

{

yylex();

}

void eval()

{

switch(oper)

{

case '+':

ans=op1+op2;

break;

case '-':

ans=op1-op2;

break;

case '\*':

ans=op1\*op2;

break;

case '/':

if(op2==0)

{

printf("ERROR");

return;

}

else

{

ans=op1/op2;

}

break;

default:

printf("operation not available");

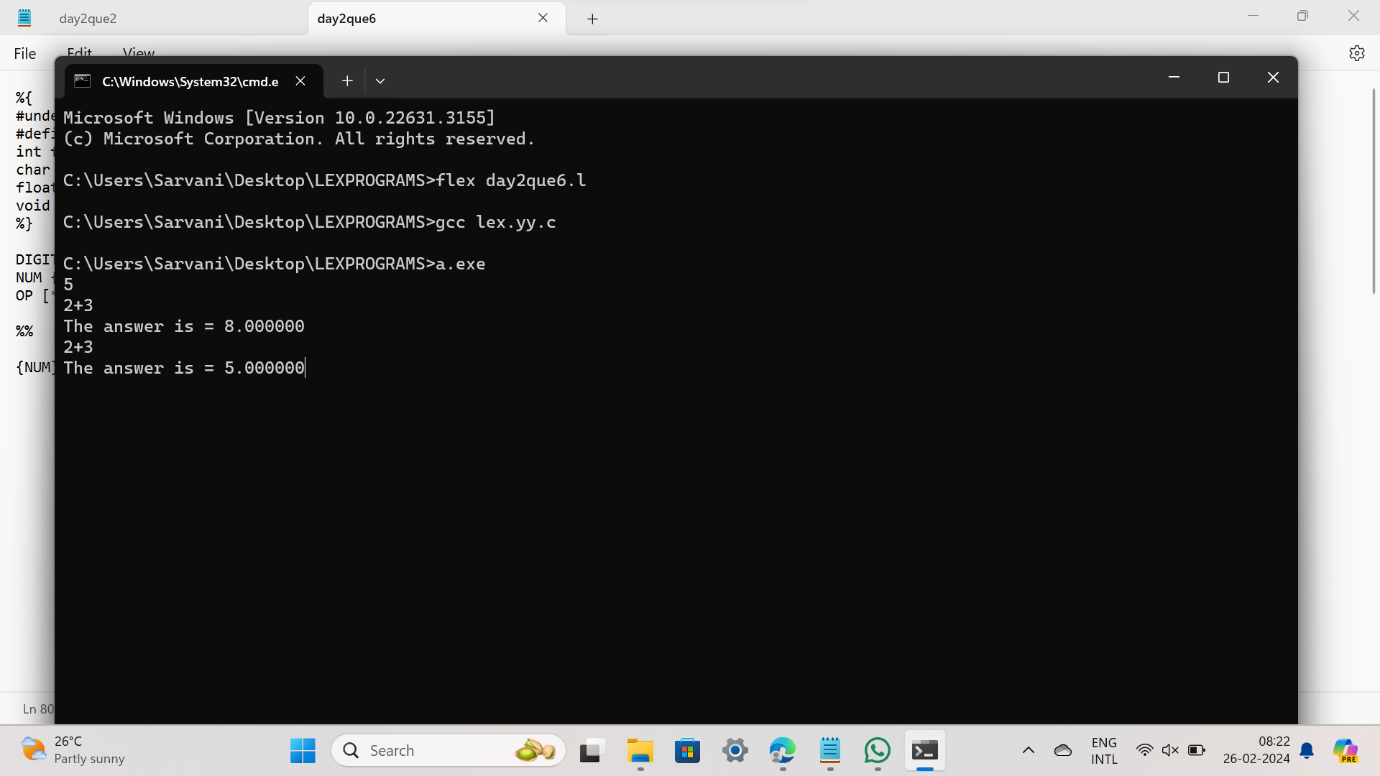
break;

}

printf("The answer is = %lf",ans);

}

**OUTPUT:**



**QUE 7**

**CODE:**

%{

int flag=0;

%}

%%

[a-z . 0-9]+@[a-z]+".com"|".in" { flag=1; }

%%

int main()

{

yylex();

if(flag==1)

printf("Accepted");

else

printf("Not Accepted");

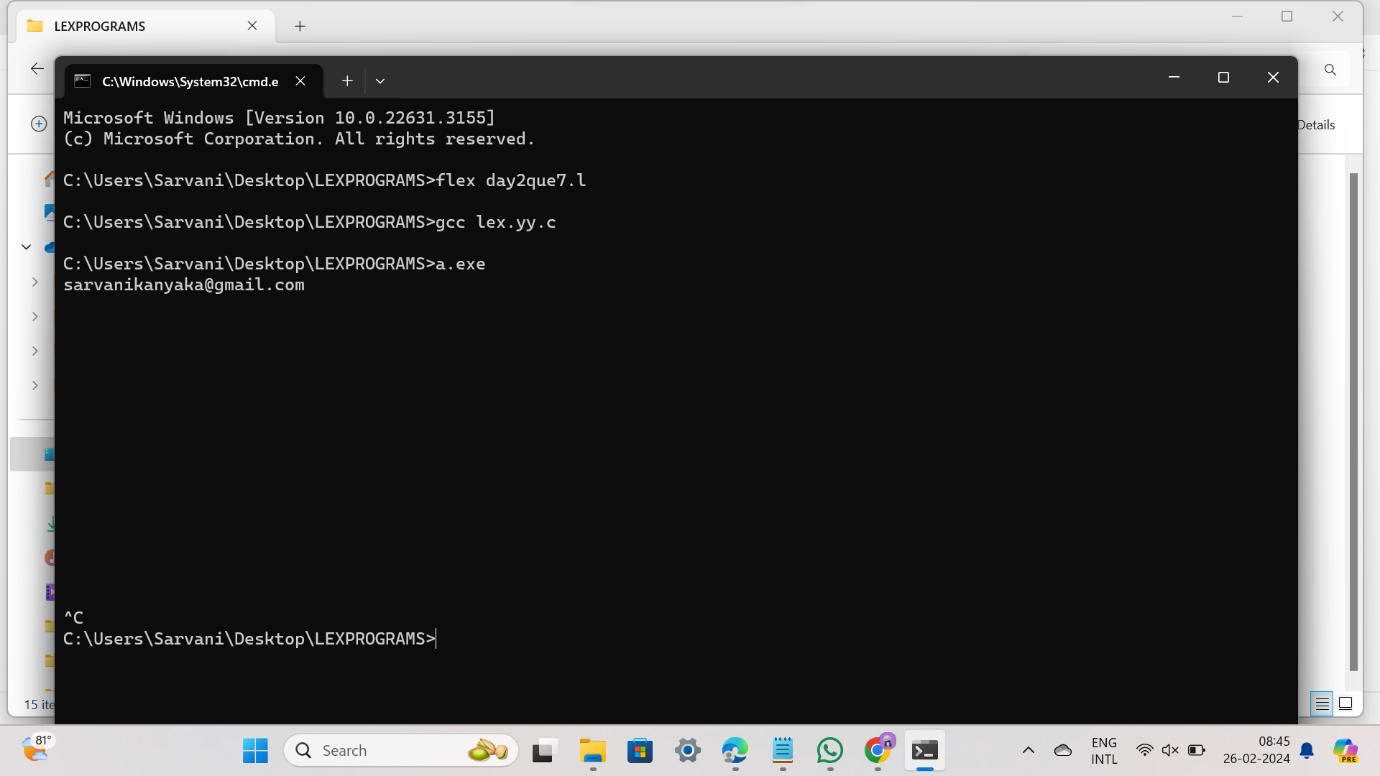
}

int yywrap()

{ return 1;

}

**OUTPUT:**



**QUE 8**

**CODE:**

%%

abc { printf("ABC"); }

. { putchar(yytext[0]); }

%%

int yywrap() {

return 0; // Indicate that there are no more input files

}

int main() {

yylex();

return 0;

}

**OUTPUT:**

