RAMANUJAN COLLEGE



(UNIVERSITY OF DELHI)

PROGRAMME NAME - B.Sc (Hons.) Computer Science

SEMESTER - V

PAPER TITLE - System Programming Practical

SUBMITTED BY – SAGAR YADAV

EXAMINATION ROLL NO. - 20020570027

COLLEGE ROLL NO. - 20201449

# PRACTICAL LIST

Ques 1. Write a Lex program to count the number of lines and characters in the input file.

CODE

%{

#include<stdio.h>

int lines=0, chars=0;

%}

%%

[\n] {lines++;}

[a-zA-Z0-9] {chars++;}

[^ \t \n]+ {chars+=yyleng;} //yyleng= length of the matched string

%%

int main()

{ yyin= fopen("Input.txt", "r"); yylex();

printf("No of lines in the file: %d", lines); printf("\n");

printf("No of characters in the file: %d", chars); printf("\n");

return 0;

}

int yywrap()

{ return 1;

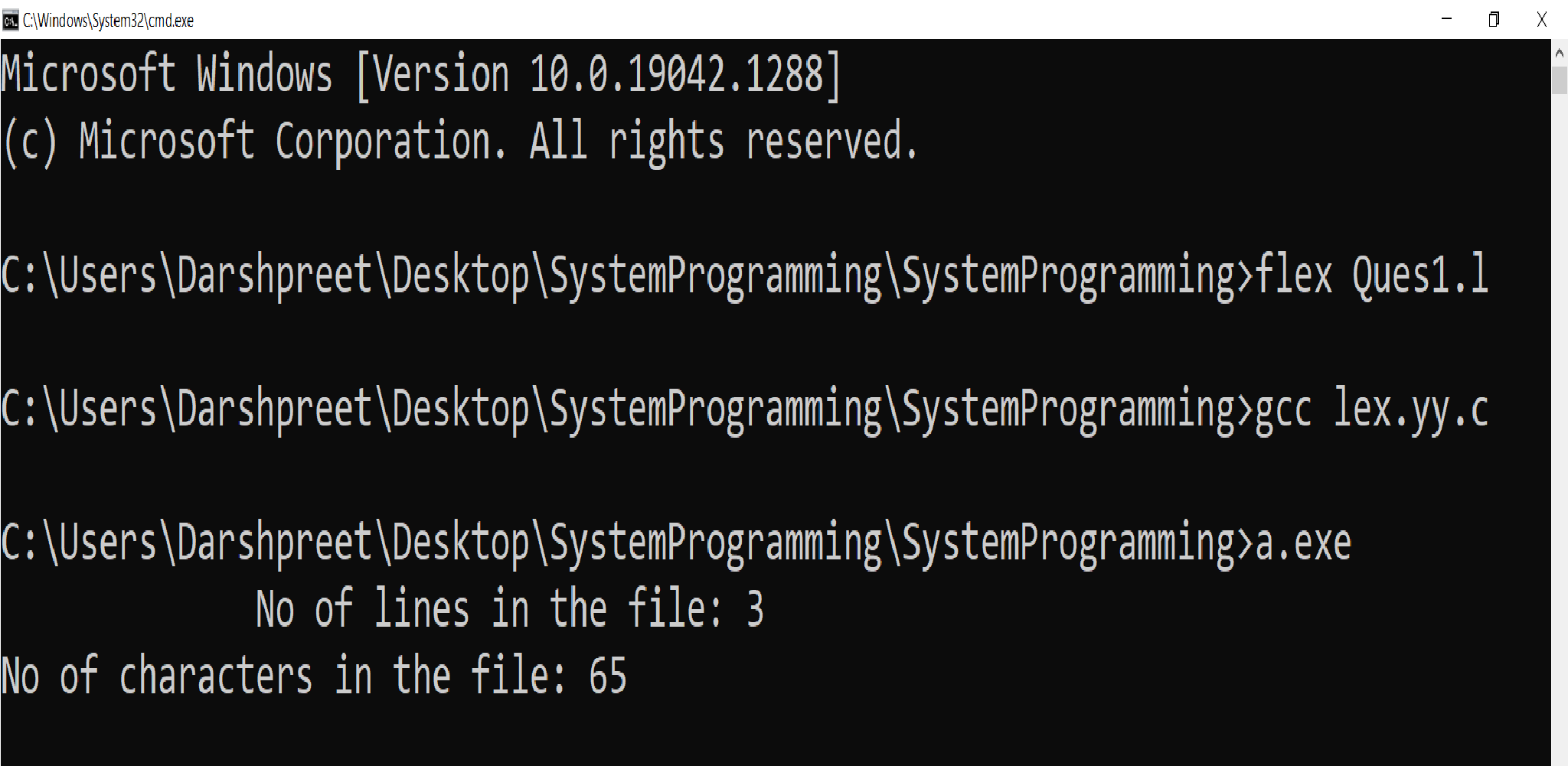
}

INPUT.txt Heyy!

My name is Darshpreet Kaur

I am a student of Mata Sundri College for Women

OUTPUT :



Ques 2. Write a Lex program that implements the Caesar cipher. It replaces every letter with the one three letters after in an alphabetical order, wrapping around at Z e.g a is replaced by d ,d bye ,and so on z by c.

CODE

%{

#include <stdio.h>

%}

%%

[a-z] {char ch=yytext[0];

ch+=3;

if(ch>'z')

ch-=('z'+1-'a');

printf("After encryption: %c", ch);

printf("\nEnter the character: ");

}

[A-Z] {char ch=yytext[0]; ch+=3;

if(ch>'Z') ch-=('Z'+1-'A'); printf("After encryption: %c", ch);

printf("\nEnter the character: ");

}

%%

int yywrap()

{ return 1;

}

int main()

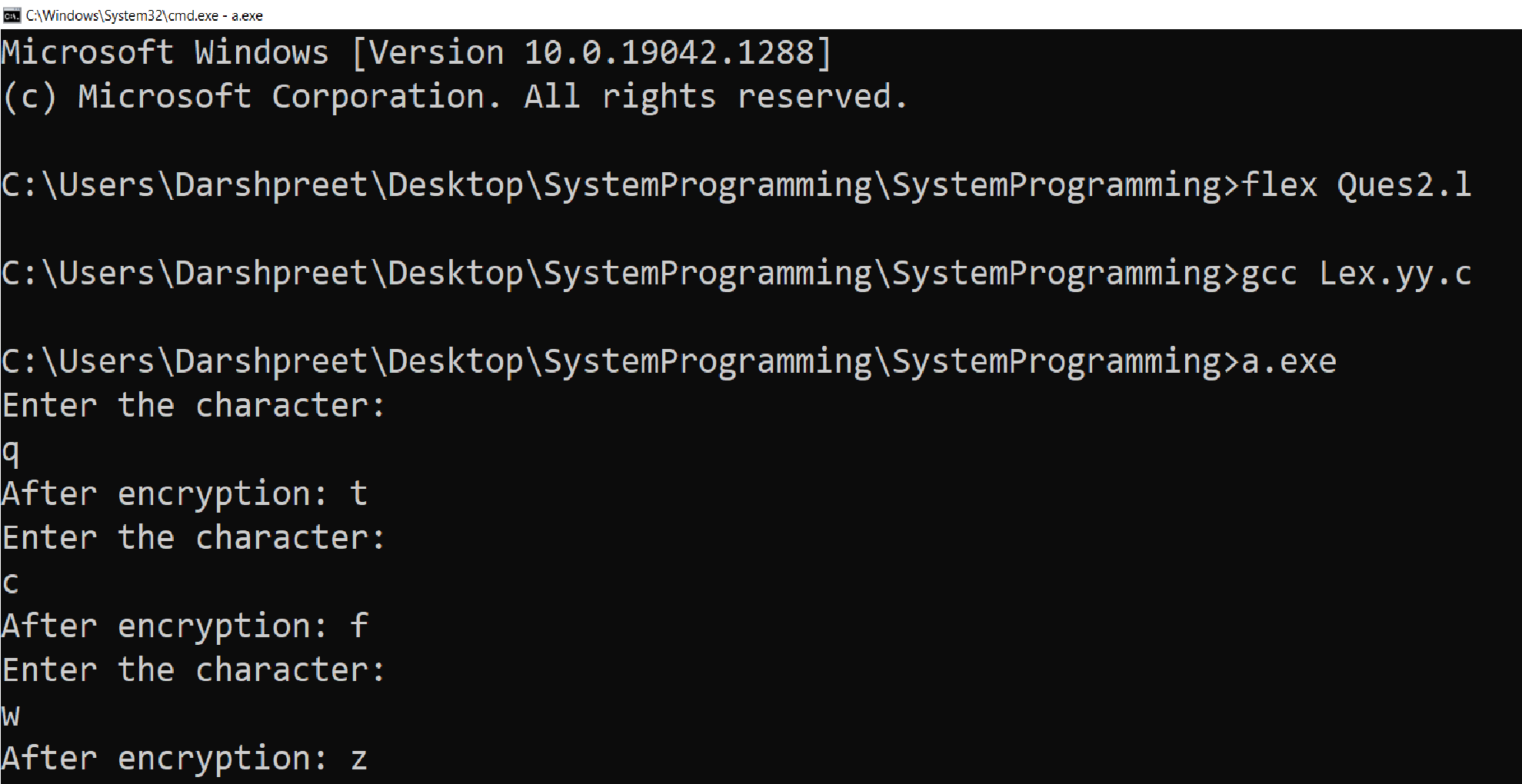
{ printf("Enter the character: \n");

yylex();

return 0;

}

OUTPUT :



Ques 3. Write a Lex program that finds the longest word (defined as a contiguous string of upper and lower case letters in the input.

CODE

%{

#include<stdio.h> #include<string.h> int count=0;

char longest[50];

%}

%%

[a-zA-Z]+ {

if(yyleng > count)

{

count= yyleng; strcpy(longest, yytext);

} }

%%

int main()

{

yyin=fopen("Input1.txt", "r"); yylex();

printf("The longest word is: %s", longest); return 0;

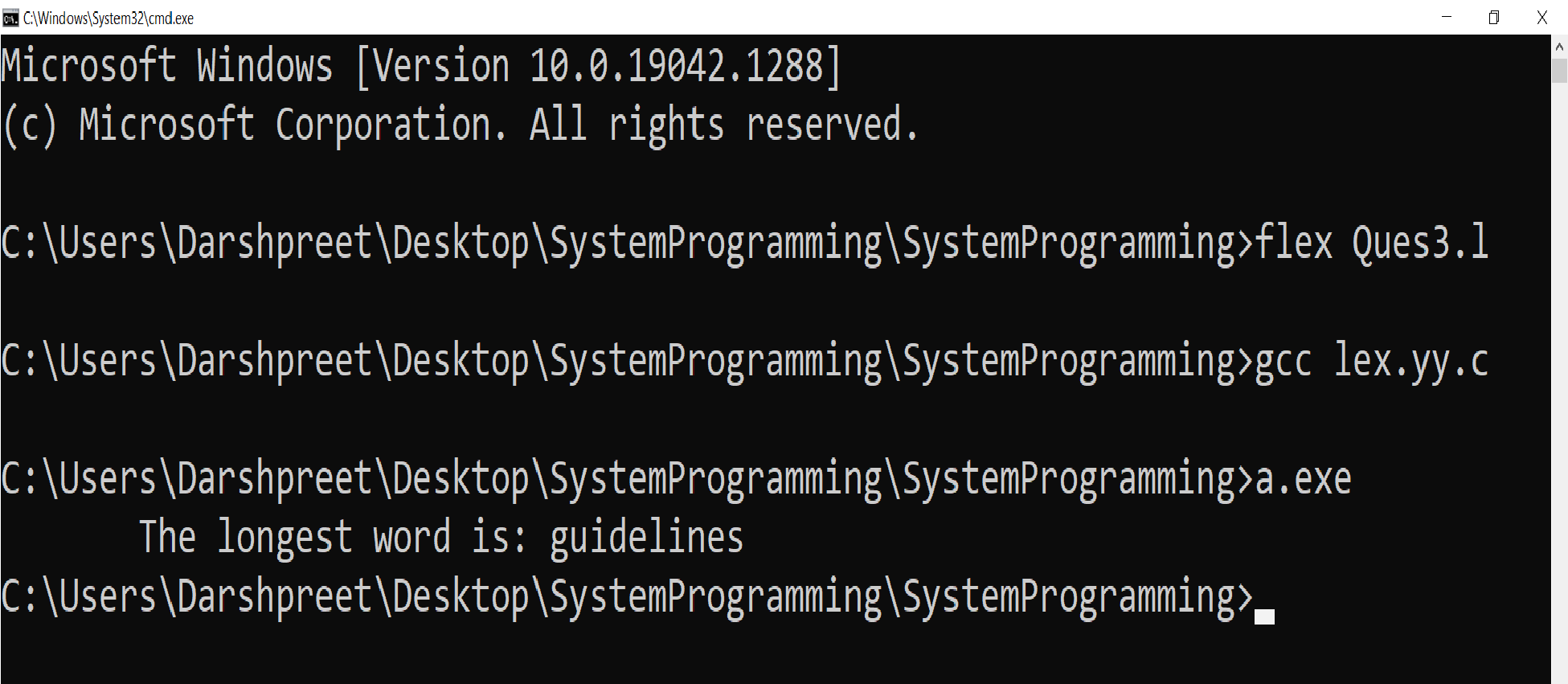
}

int yywrap() {return 1;}

Input1.txt

Problems are not stop signs they are guidelines

OUTPUT :



Ques 4. Write a Lex program that distinguishes keywords, integers, floats, identifiers, operators and comments in any simple programming language.

CODE

%{

#include<stdio.h>

%}

digit[0-9] letter[A-Za-z\_] opr[+ - \* /]

%%

" "|"\t";

{digit}+ {printf("Integer");}

{digit}+\.{digit}+ {printf("Float");} {opr} {printf("Operator");} if|else|while|switch|for|char {printf("Keyword");}

{letter}({letter}|{digit})\* {printf("Identifier");}

[/]{1}[/]{1}[a-zA-Z0-9]\* {printf("Single Line Comment");}

[/]{1}[\*]{1}[a-zA-Z0-9]\*[\*]{1}[/]{1} {printf("Multi Line Comment");}

.\* printf("Invalid");

%%

int main()

{ yylex();

return 0;

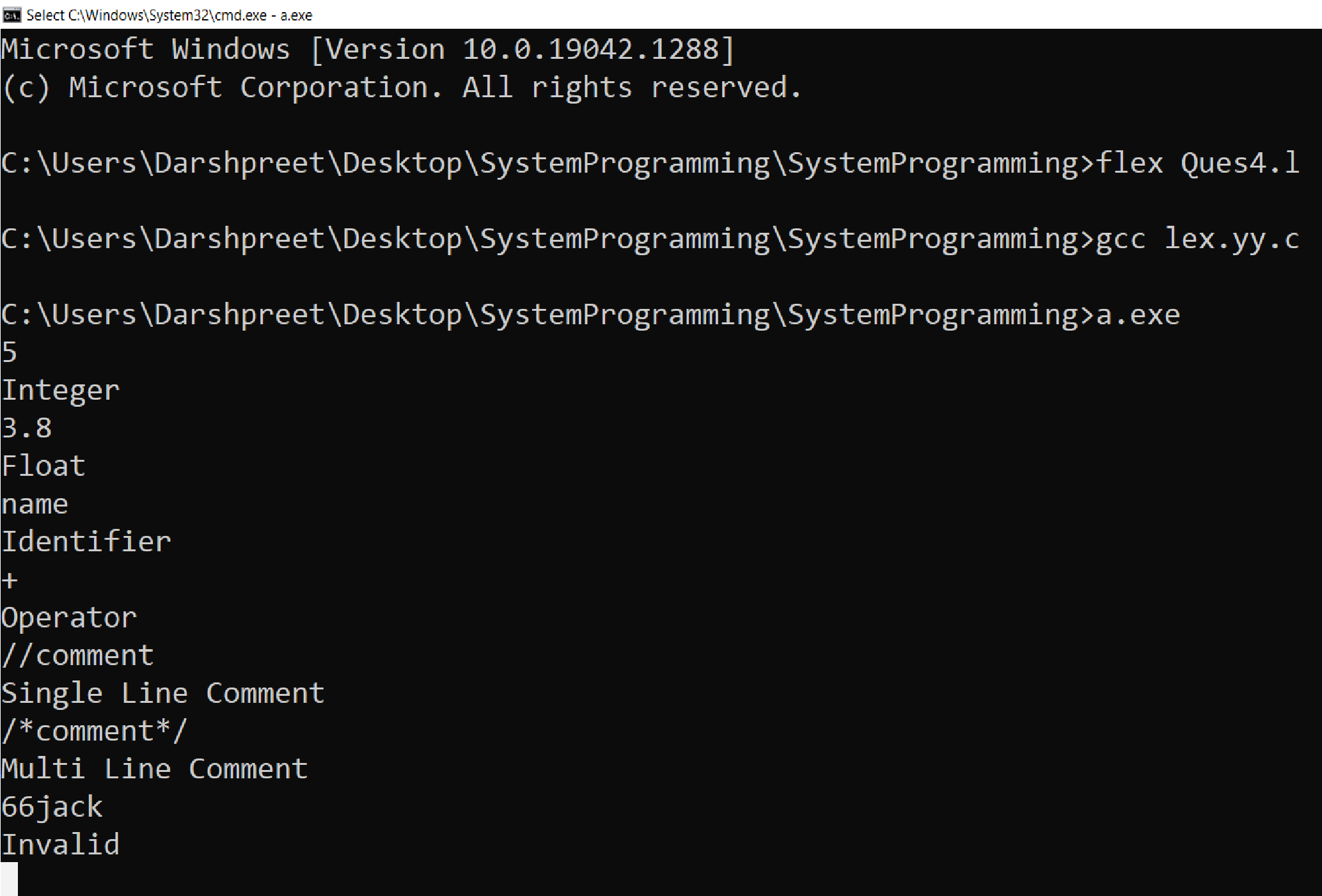
}

int yywrap()

{ return 1;

}

OUTPUT :



Ques 5. Write a Lex program to count the number of identifiers in a C file.

CODE

%{

char ch;

int id;

%}

%%

^[ \t]\*(int|float|double|char) { ch=input();

while(1)

{ if(ch==',') id++; else if(ch==';')

{ id++;

break;

}

ch=input();

} }

.|[\n] ;

%%

int yywrap(){

return 1;

}

int main()

{ yyin=fopen("Input.c","r"); yylex(); printf("\nTotal identifiers is %d\n",id); }

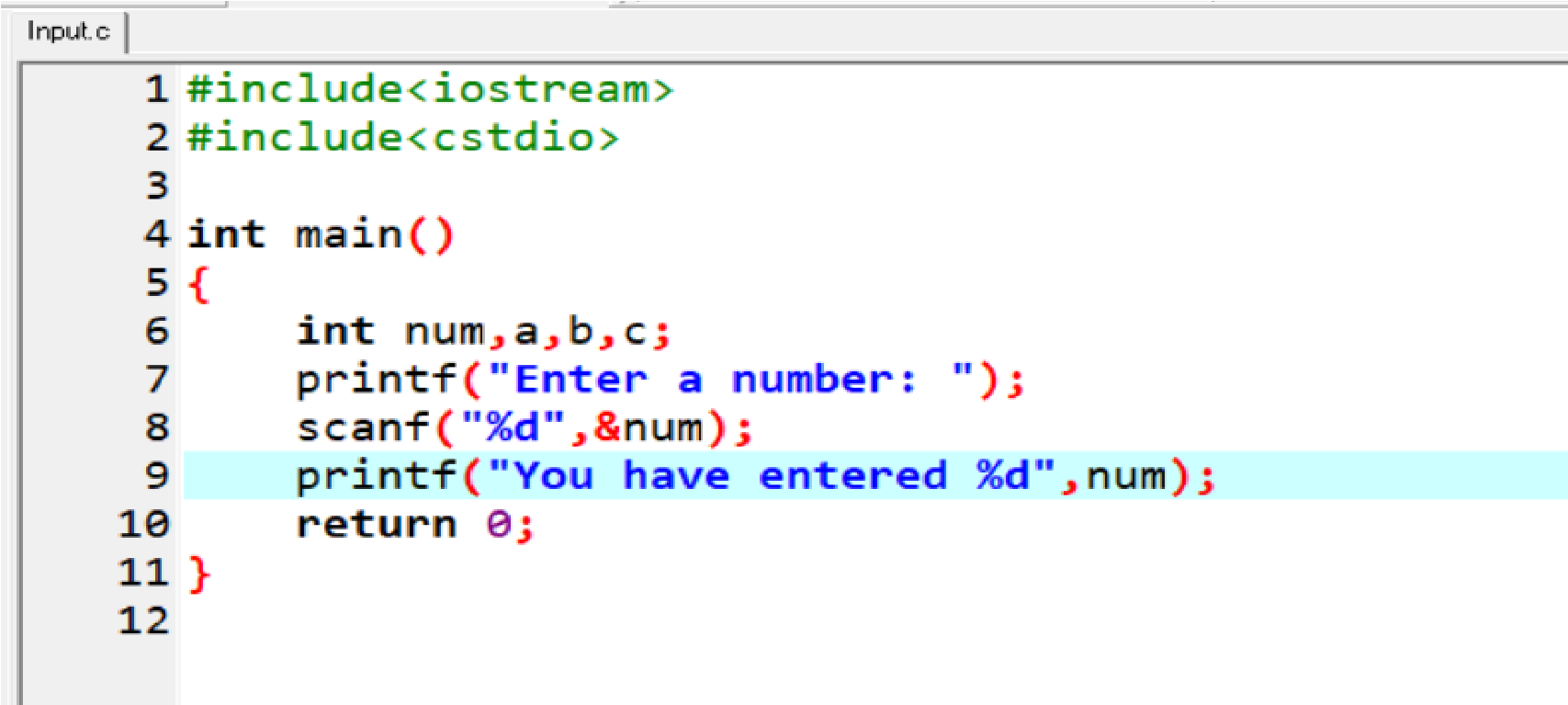
Input.c

#include<iostream> #include<cstdio>

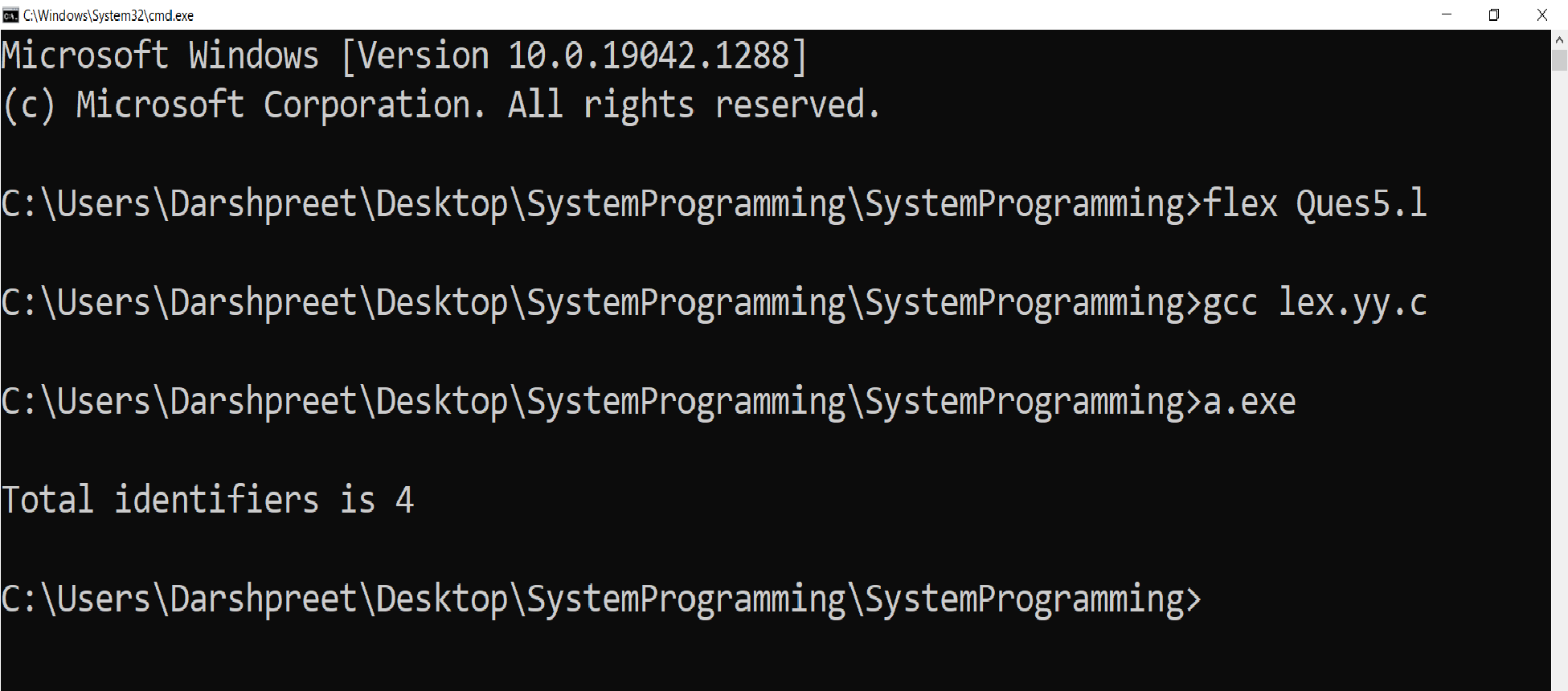
int main()

{ int num,a,b,c; printf("Enter a number: "); scanf("%d",&num); printf("You have entered %d",num); return 0;

}



OUTPUT :



Ques 6. Write a Lex program to count the number of words, characters, blank spaces and lines in a C file.

CODE

%{

#include<stdio.h>

int lines =0, chars= 0, spaces=0, words=0;

%}

%%

[\n] {lines++;}

[ ]|[\t] {spaces++;}

[^ \t \n]+ {words++; chars+=yyleng;}

%%

int main()

{ yyin= fopen("Input.c", "r"); yylex();

printf(" This File contains ..."); printf("\n");

printf("No of lines in the file: %d", lines); printf("\n");

printf("No of spaces in the file: %d", spaces); printf("\n");

printf("No of characters in the file: %d", chars); printf("\n");

printf("No of words in the file: %d", words); printf("\n");

return 0;

}

int yywrap() {return 1;}

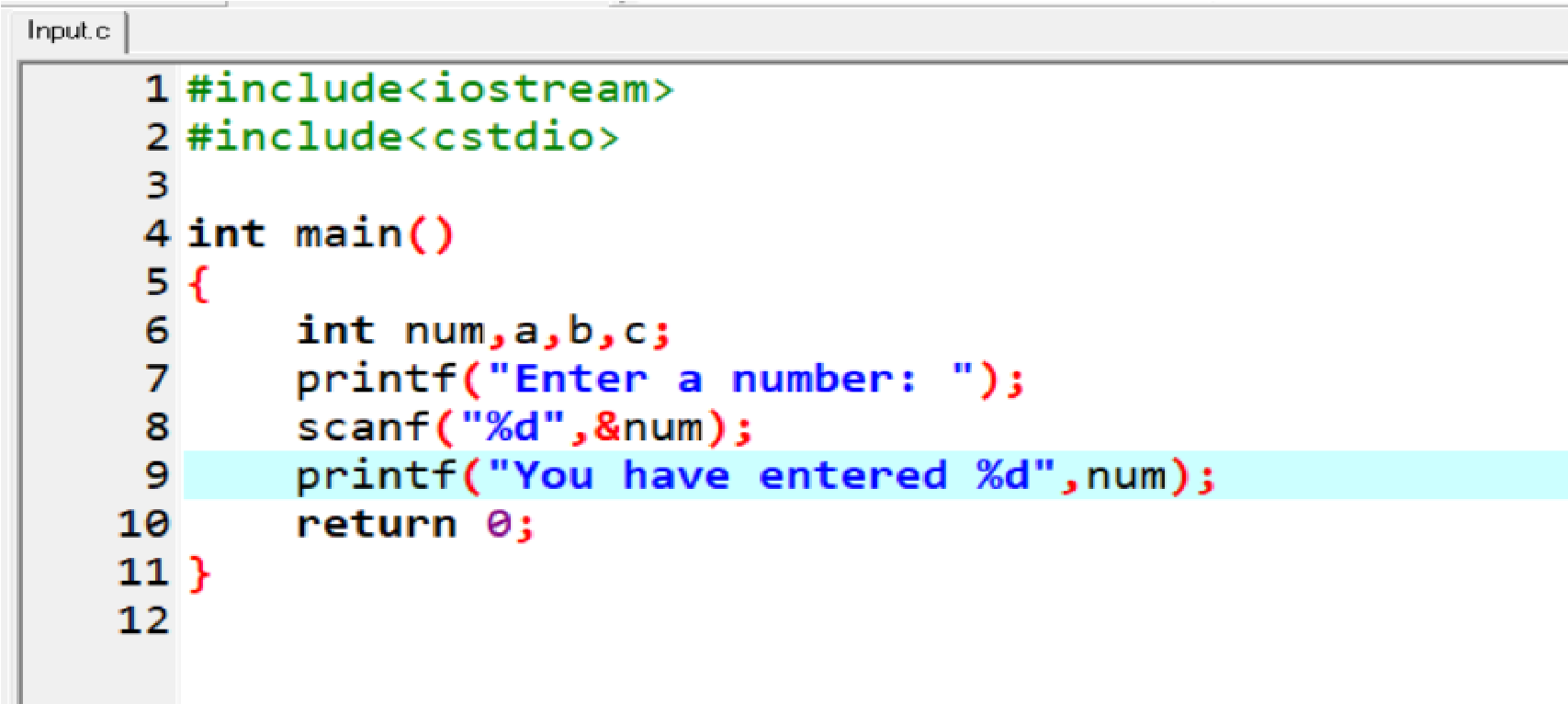
Input.c

#include<iostream> #include<cstdio>

int main()

{ int num,a,b,c; printf("Enter a number: "); scanf("%d",&num); printf("You have entered %d",num); return 0;

}



OUTPUT :



Ques 7. Write a Lex specification program that generates a C program which takes a string “abcd” and prints the following output.

abcd abc ab a

CODE

%{

#include<stdio.h> #include<string.h> char ch[8];

int i,j;

%}

char [a-zA-Z]

%%

{char}+ { printf("\n");

for(i=yyleng;i>=0;i--)

{ for( j=0;j<i;j++) printf("%c",yytext[j]);

printf("\n");

} }

%%

int yywrap()

{ return 1;

}

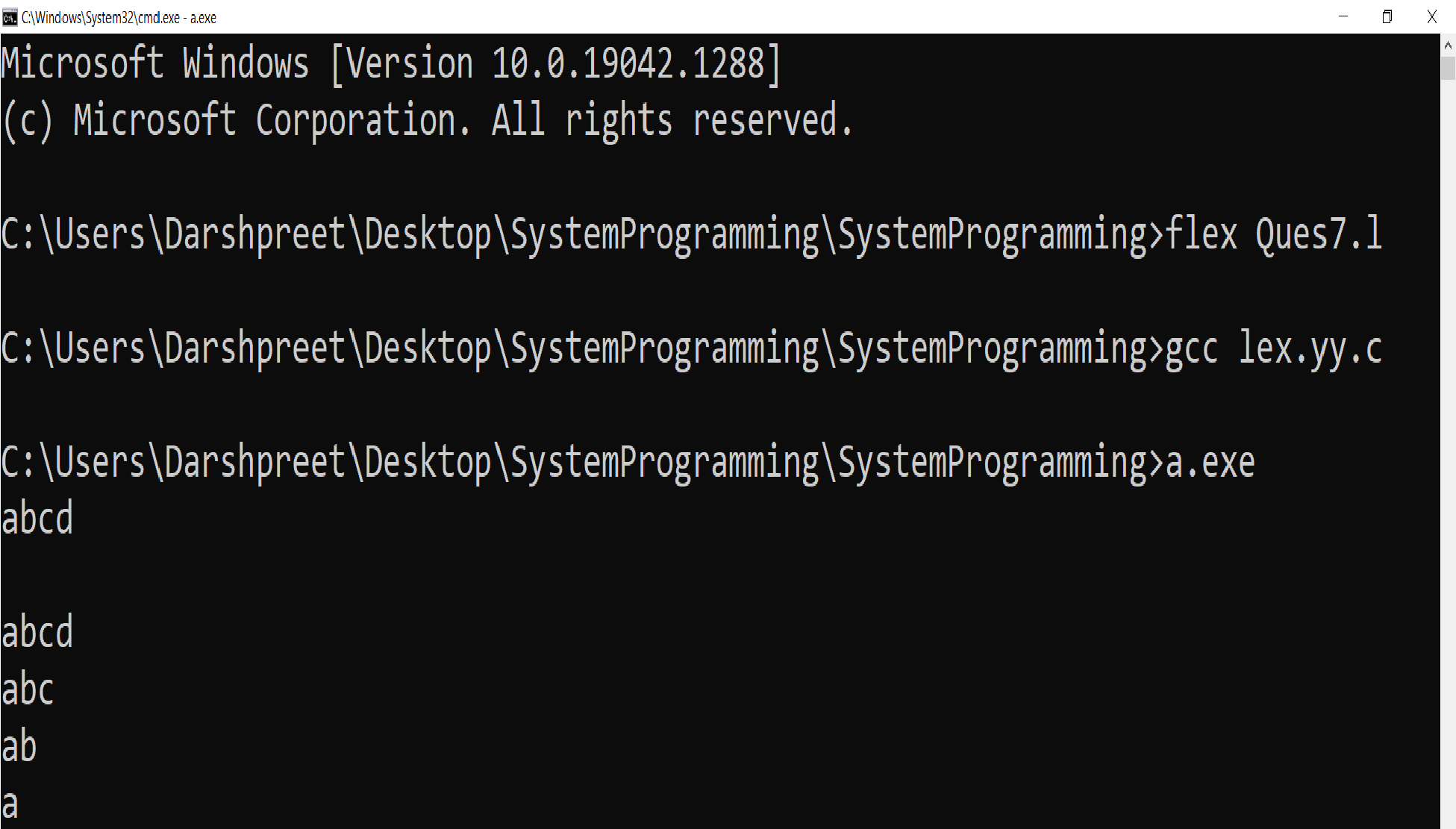
int main()

{ yylex();

return 0;

}

OUTPUT :



Ques 8. A program in Lex to recognize a valid arithmetic expression.

CODE

%{

#include<stdio.h> #include<stdlib.h>

int c=0,d=0,bo=0,bc=0;

%}

operand [a-zA-Z0-9]+ operator [+\-\/\*]

%%

{operator} {d++;

printf("%s is an operator \n",yytext);}

{operand} {c++;

printf("%s is an operand \n",yytext);}

"(" {if(bc<=bo) bo++;}

")" {bc++;}

\n {if(bo==bc&&c>d){printf("valid expression");} else{printf("invalid expression");} exit(0);}

%%

int yywrap()

{

return 0;

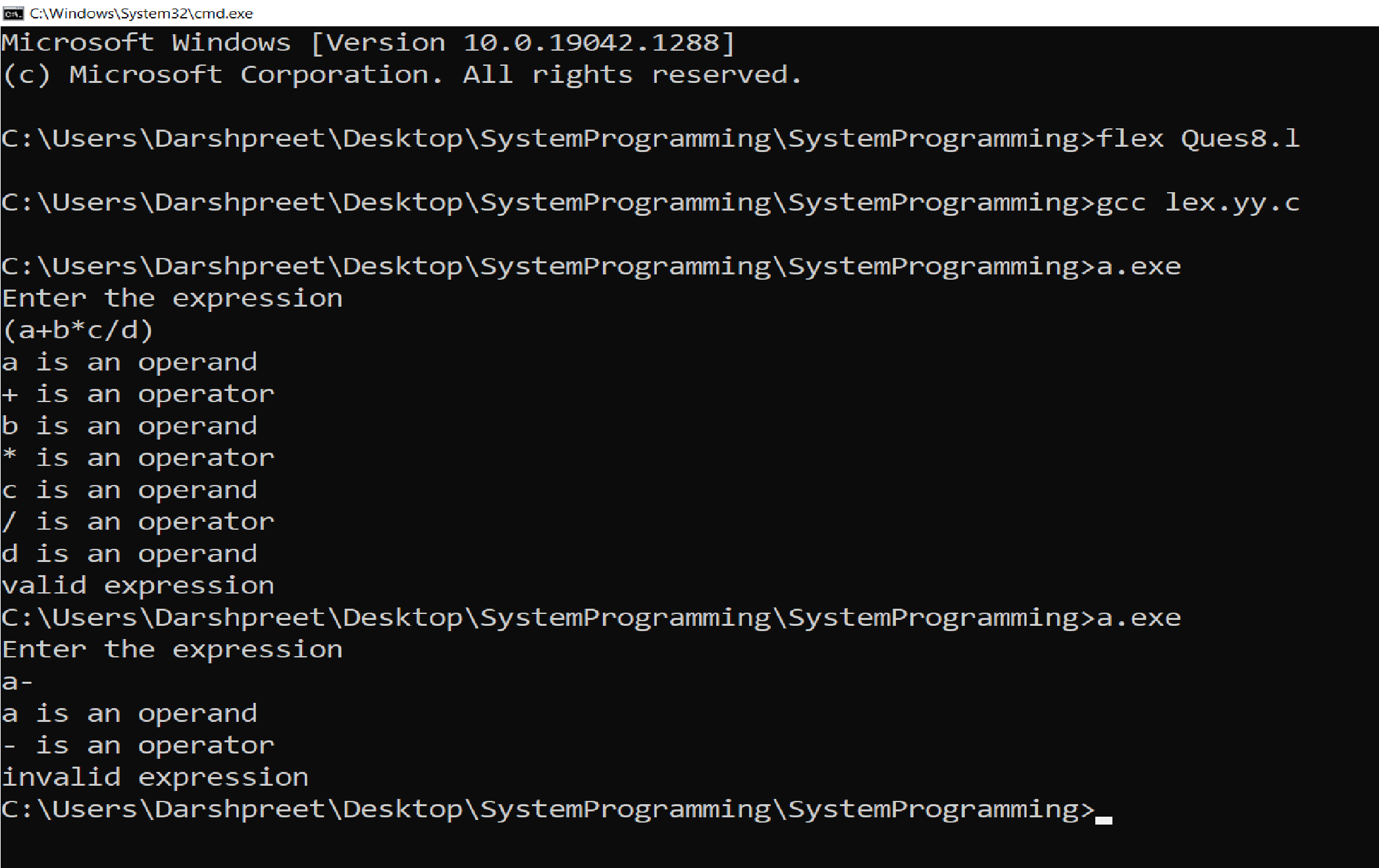
}

void main()

{ printf("Enter the expression\n"); yylex();

}

OUTPUT :



Ques 9. Write a YACC program to find the validity of a given expression (for operators +-\* and/).

CODE (L file)

%{

#include "Ques9.tab.h" extern yylval;

%}

/\* defined section \*/

%%

[0-9]+ {yylval=atoi(yytext); return NUMBER;} //this is send to the yacc code as token INTEGER

[a-zA-Z]+ {return ID;} //this is send to the yacc code as token ID [\t]+ ;

\n {return 0;}

. {return yytext[0];}

%%

int yywrap()

{ return 1;

}

CODE (Y file) %{

#include<stdio.h>

int valid=1;

%}

//definition section

%token NUMBER ID // token from lex file

%left '+' '-' // left associative

%left '\*' '/'

%%

expr: expr '+' expr // grammer production rule

|expr '-' expr

|expr '\*' expr

|expr '/' expr

|'-'NUMBER

|'-'ID

|'('expr')'

|NUMBER

|ID

;

%%

//main function

main()

{

printf("Enter the expression\n"); yyparse();

if(valid)

printf("\nExpression is valid\n");

}

//if error occured

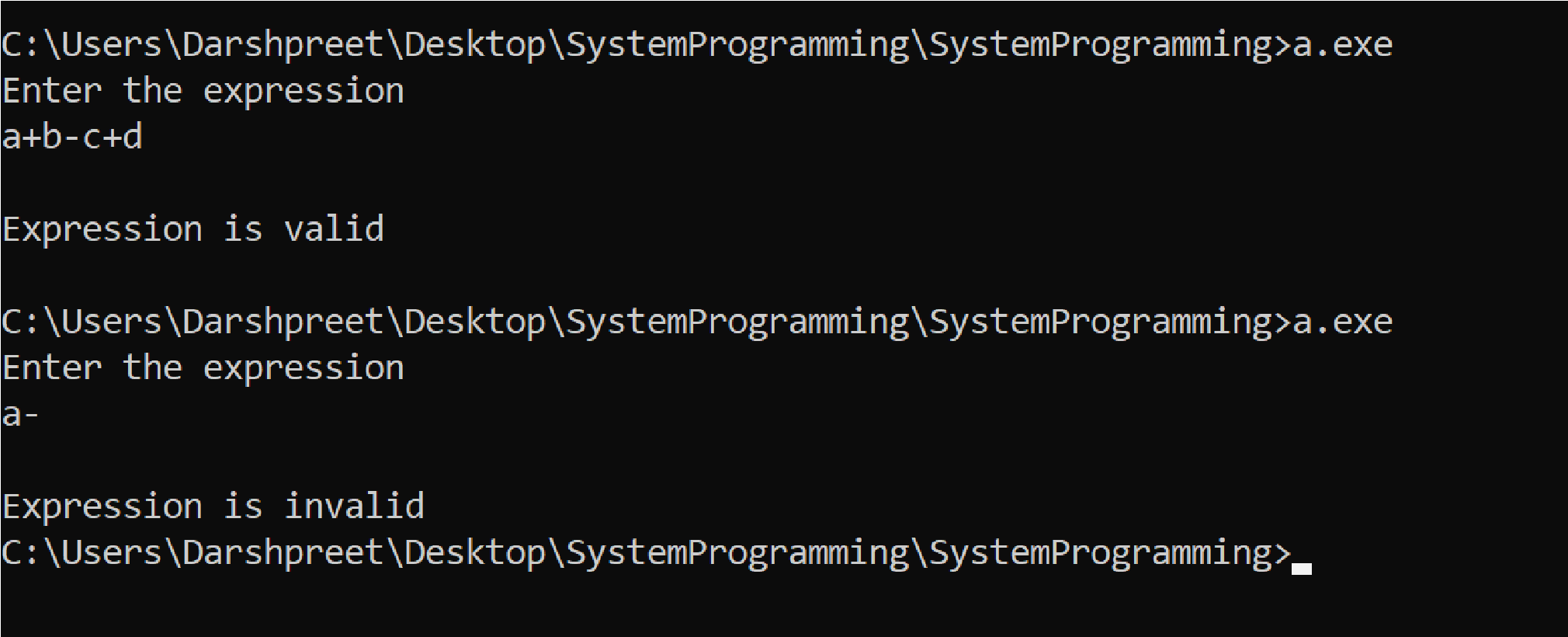
int yyerror(char \*s)

{

valid=0;

printf("\nExpression is invalid"); }

OUTPUT :



Ques 10. A Program in YACC which recognizes a valid variable which starts with a letter followed by a digit. The letter should be in lowercase only.

CODE (L file)

%{

#include "Ques10.tab.h"

%}

letter [a-z] digit [0-9] newline [\n]

%%

{letter} { return letter ;}

{digit} { return digit ; }

{newline} { return newline ;}

. { printf("Invalid Variable\n");exit(0); }

%%

int yywrap()

{ return 1; }

CODE (Y file) %{

#include<stdio.h>

#include<stdlib.h>

%}

%token letter digit newline

%%

line: variable newline {printf("Valid variable!!\n"); exit(0);}; variable : letter term {printf("variable letter\n");};

term: letter term {printf("letter term\n");}| digit {printf("digit\n");};

%%

int yyerror(char \*msg)

{ printf("Invalid variable!!\n"); exit(0);

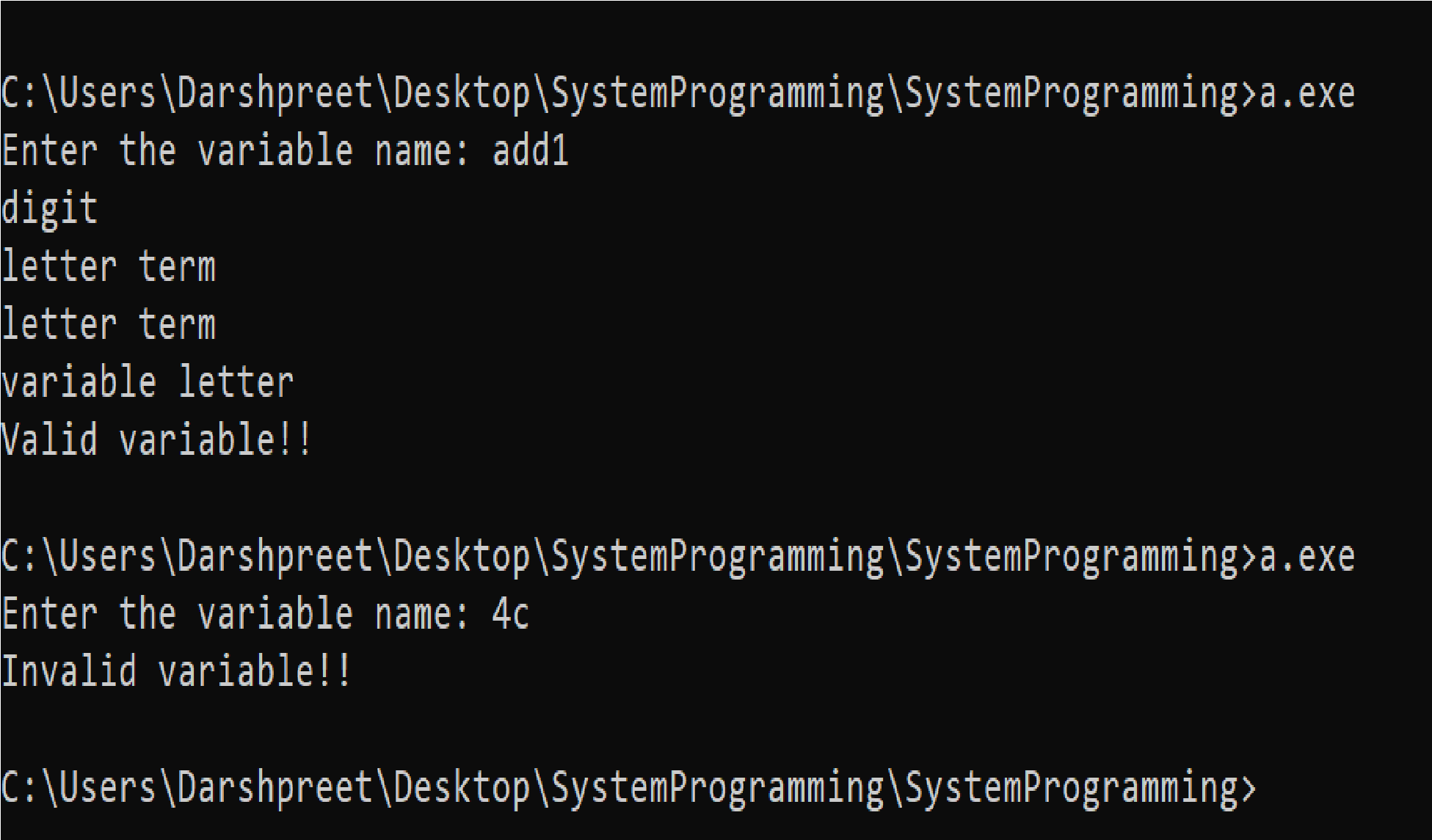
}

int main ()

{ printf("Enter the variable name: "); yyparse();

}

OUTPUT :



Ques 11. A Program in YACC to evaluate an expression (simple calculator program for addition and subtraction, multiplication, division).

CODE (L file)

%{

#include "Ques11.tab.h"

extern int yylval;

%}

%%

[0-9]+ { yylval=atoi(yytext);return NUMBER;}

\n {return 0;}

. {return yytext[0];}

%%

int yywrap()

{ return 1; }

CODE (Y file) %{

#include<stdio.h> int yylex(void);

int yyerror(char \*);

%}

%token NAME NUMBER

%%

statement : NAME '=' expression | expression { printf("=%d\n",$1);}

;

expression:expression'+' NUMBER{$$ = $1+$3;}

|expression '-' NUMBER {$$ = $1-$3;}

|expression '\*' NUMBER {$$ = $1\*$3;}

|expression '/' NUMBER { if ($3!=0){ $$ = $1/$3; }else { printf("Error: divide by Zero"); }

}

|NUMBER {$$=$1;}

;

%%

int main()

{ yyparse();

return 0;

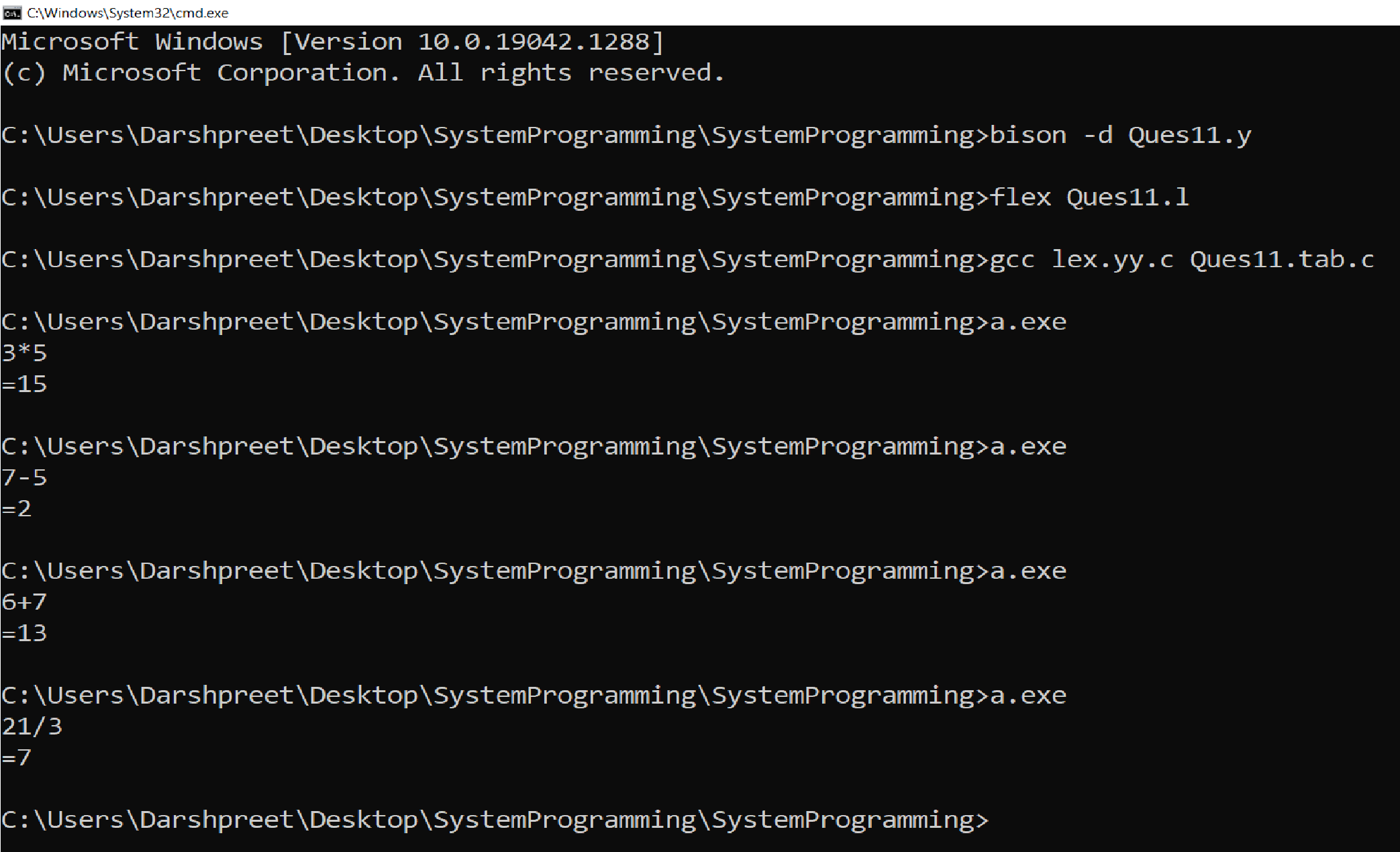
}

int yyerror(char \*s)

{ printf("%s",s);

}

OUTPUT :



Ques 12. A Program in YACC to recognise the strings “ab” , “abab” , “ababab”\_\_\_\_ of the language (anbn , n>=1).

CODE (L file)

%{

#include "Ques12.tab.h"

%}

%%

a return A; b return B; .|[\n] return 0;

%%

int yywrap()

{ return 1;

}

CODE (Y file) %{

#include<stdio.h>

int valid=1;

%}

%token A B

%%

start : A start B

| ;

%%

int yyerror()

{

valid=0; printf("\nInvalid pattern!!\n");

return 0;

}

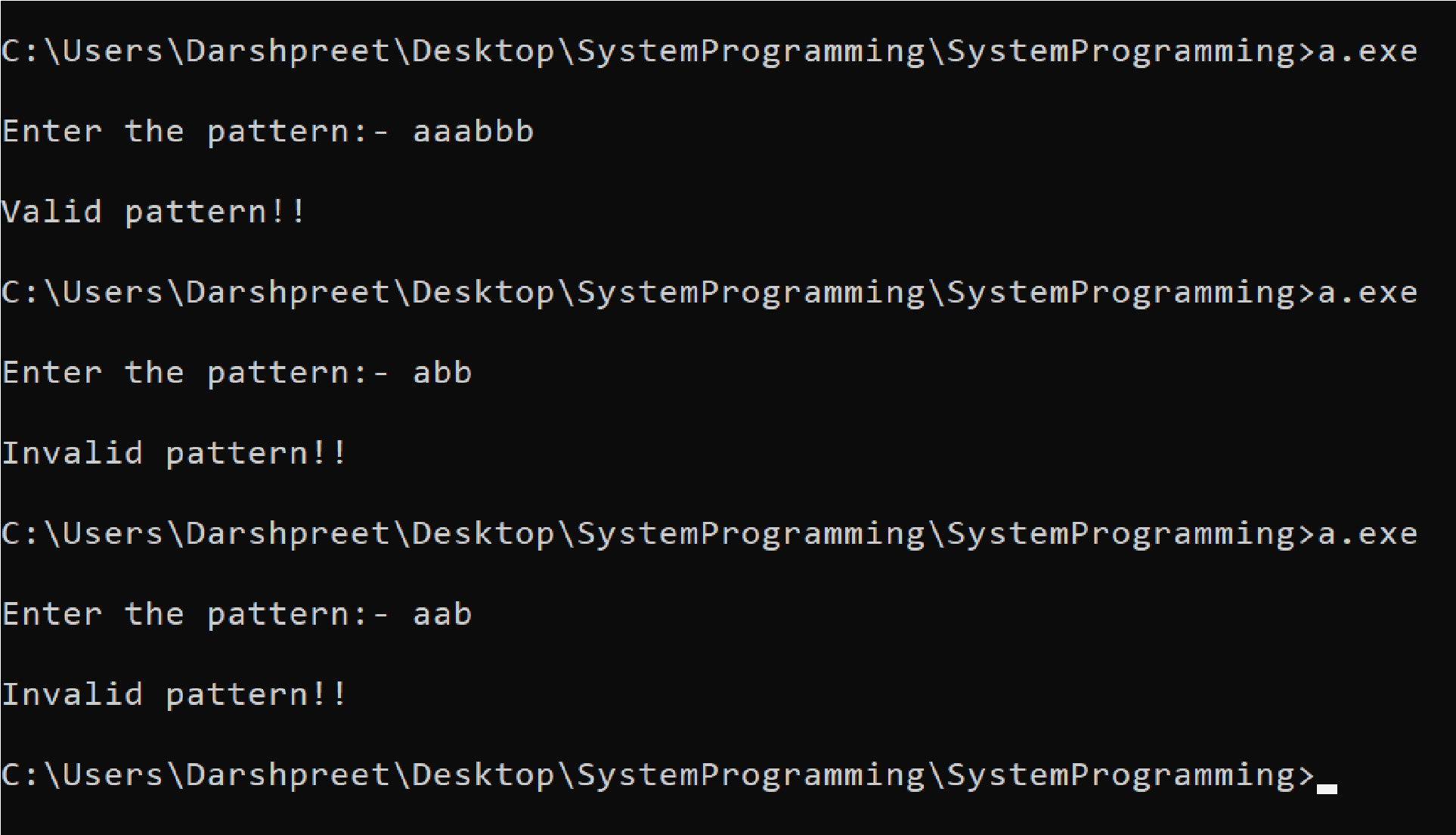
int main()

{ printf("\nEnter the pattern:- "); yyparse(); if(valid)

{ printf("\nValid pattern!!\n");

} }

OUTPUT :



Ques 13. A Program in YACC to recognize the language (𝑎𝑛𝑏 , n>=10).

(Output to say input is valid or not).

CODE (L file)

%{

#include "Ques13.tab.h"

%}

%%

[aA] {return A;}

[bB] {return B;}

\n {return NL;}

. {return yytext[0];}

%%

int yywrap()

{ return 1; }

CODE (Y file) %{

#include<stdio.h>

#include<stdlib.h>

%}

%token A B NL

%%

stmt: S B NL {printf("Valid string\n");

exit(0);}

;

S: A A A A A A A A A A X

;

X: A X

| ;

%%

int yyerror(char \*msg)

{ printf("Invalid String!!!\n"); exit(0);

}

main()

{ printf("Enter the string\n"); yyparse();

}

OUTPUT :

