VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

Lab REPORT
on

Compiler Design

Submitted by

Jigar D Patel (1BM21CS081)

Under the Guidance of
Prof. Sunayana S
Assistant Professor, BMSCE

in partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING BENGALURU-560019
Nov-2023 to Feb-2024
(Autonomous Institution under VTU)

B. M. S. College of Engineering,

Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the lab work entitled "**Compiler Design**" carried out by **Jigar D Patel (1BM21CS081)** who are bona fide students of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visveswaraiah Technological University, Belgaum during the year 2023-2024. The lab report has been approved as it satisfies the academic requirements in respect of compiler design lab **(22CS5PCCPD)** work prescribed for the said degree.

Sunayana S
Assistant Professor
Dept. of CSE
BMSCE, Bengaluru

Dr. Jyothi S Nayak
Prof.& Head
Dept. of CSE
BMSCE, Bengaluru

INDEX

NAME: Jigar.D.Patel STD BMS SEC: B ROLL NO: C8081

S.No.	Date	Title	Page No.	Teacher's Sign/ Remarks
1	6/11/23	Program to design Lexical Analyses to identify separate keywords, identifiers		
2.	20/11/23	Count the Number of vowels and consonants.		
3	27/11/23	Floating point Numbers.		
4.	4/12/23	Replacing sequence of non-empty spaces with single space		
5	11/12/23	Recognize tokens over alphabets {0, ... 8, 9}		
6.	18/12/23	Program to design Lexical Analyzer		
7.	8/1/24	Recursive descent Program		
8	8/1/24	Design Parsing using YACC		
9.	29/1/24	YACC prog to gen. Syntax tree for a given arithmetic expression		
10	29/1/24	Infix to Prefix using YACC		
11	29/1/24	YACC generate 3-address code		
12.	29/1/24	YACC program using grammar rules		

Bafna Gold

Code:

4) write a lex program to identify each character as consonant or vowel in a given sentence.

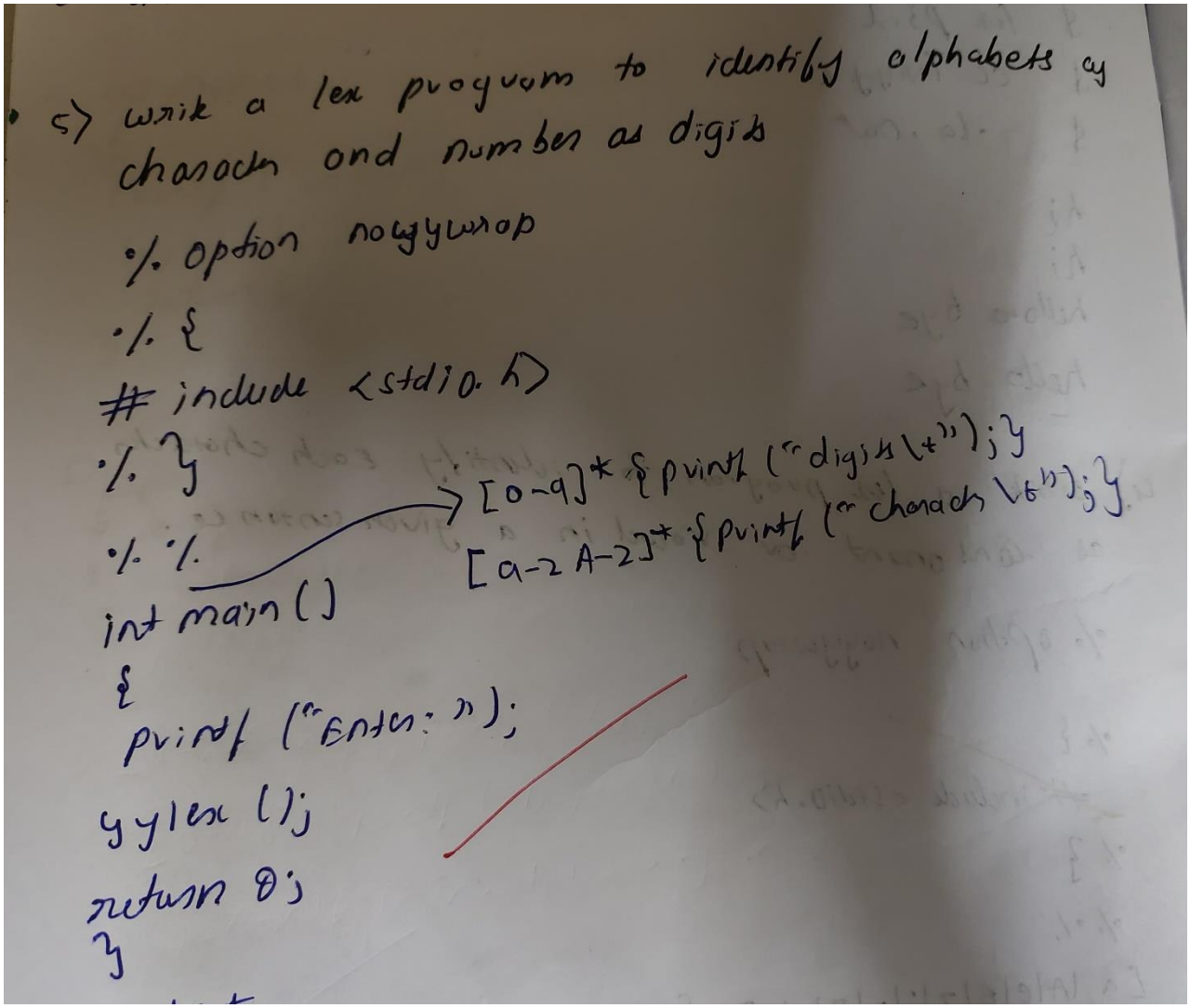
```
%option noyywrap

%{
#include <stdio.h>
%}

%%
[a|A|e|E|I|i|o|O|u|U] {printf("vowel\t");}
[a-zA-Z] {printf("consonant\t");}
%%
```

```
int main()
{
  printf("enter");
   yylex();
   return 0;
}
```

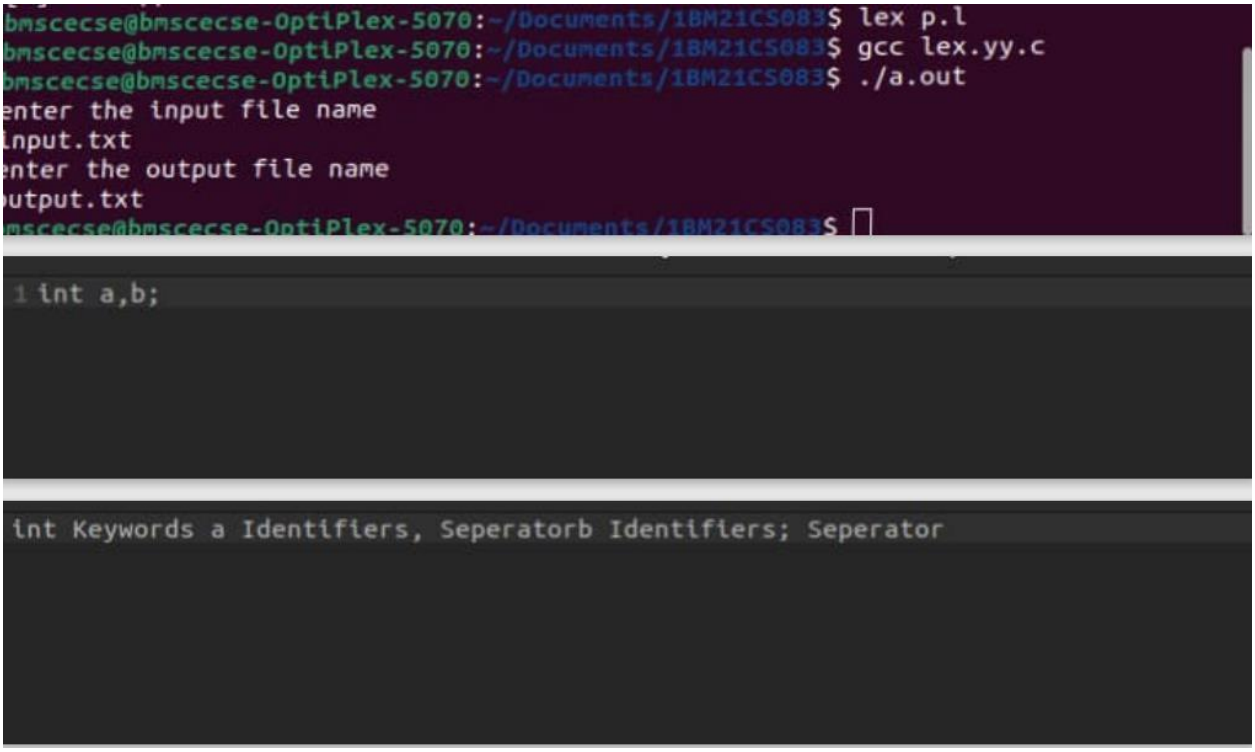
Output:

```
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ lex p4.l
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ gcc lex.yy.c
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
abcdef
vowel:a
consonant:b
consonant:c
consonant:d
vowel:e
consonant:f
number of vowels 2
number of consonants 4
```

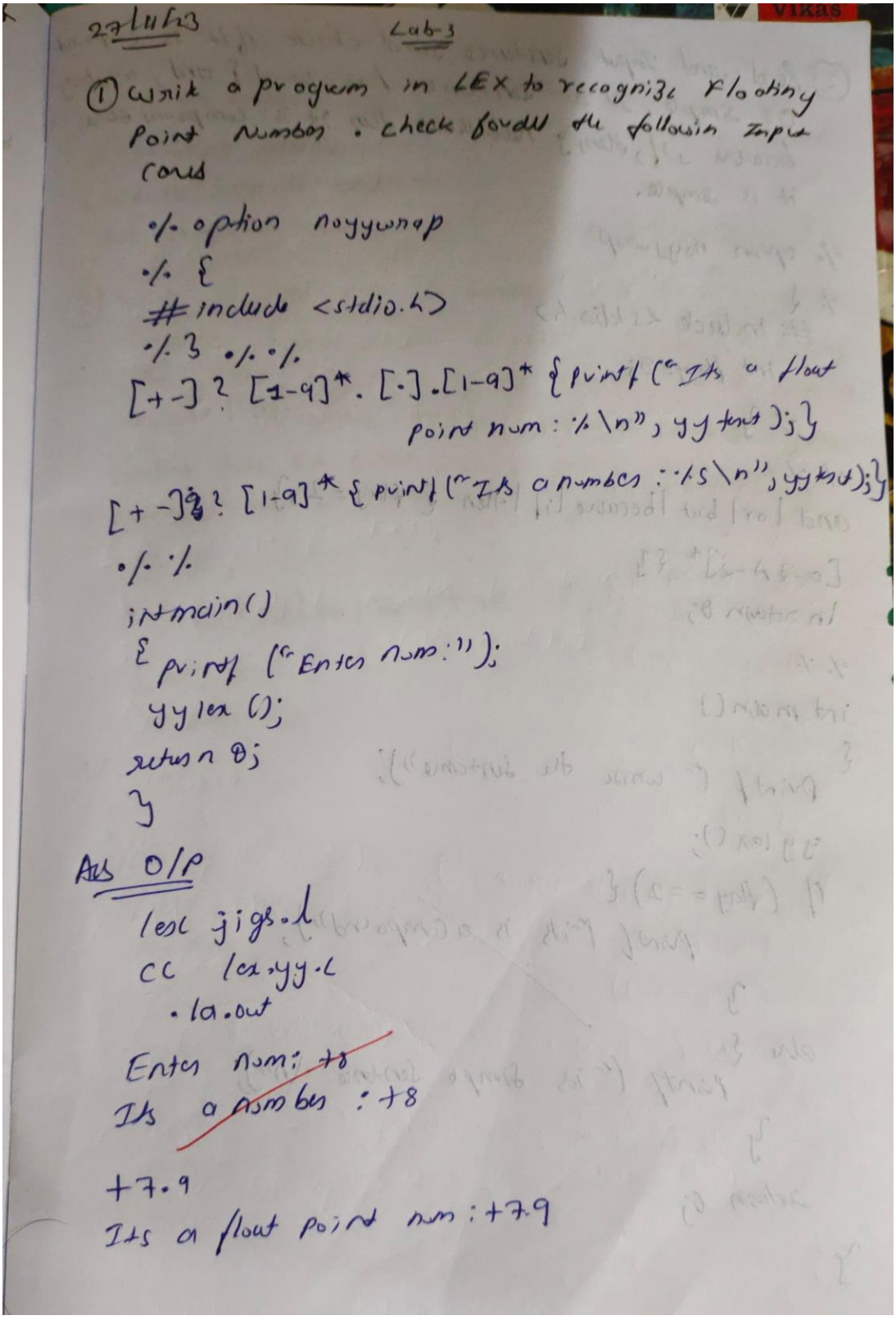
Code:


5) write a lex program to identify alphabets as
characters and numbers as digits
%option noyywrap
%{
#include <stdio.h>
%}
%%
[0-9]* {printf("digits\t");}
[a-zA-Z]* {printf("characters\t");}
int main()
{
printf("Enter:");
yylex();
return 0;
}

Output:


bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ lex p.l
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ gcc lex.yy.c
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
enter the input file name
input.txt
enter the output file name
output.txt
1 int a,b;
int Keywords a Identifiers, Seperatorb Identifiers; Seperator

Code:



27/11/23 Lab-3

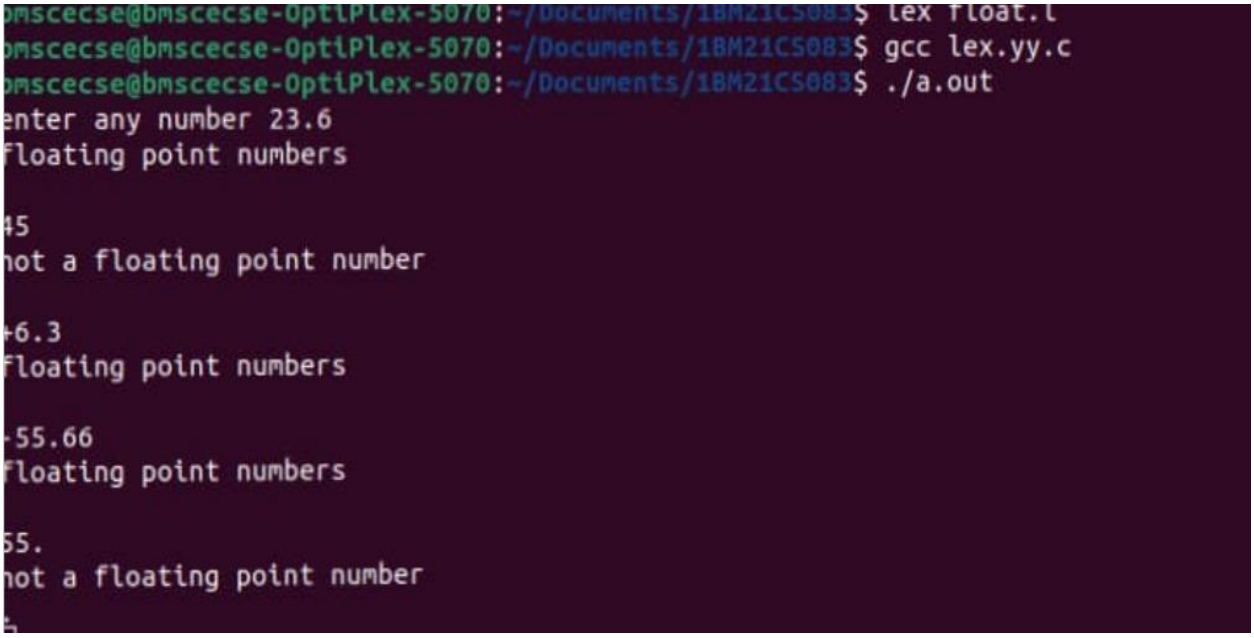
① Write a program in LEX to recognize floating point numbers. check for all the following inputs

```
%option noyywrap
%{
#include <stdio.h>
%}
%%
[+-]?[1-9]*.[0].[1-9]* {printf("Its a float point num: %s\n", yytext);}
[+-]?[1-9]* {printf("Its a numbers: %s\n", yytext);}
%%
int main()
{printf("Enter num:");
yylex();
return 0;
}
```

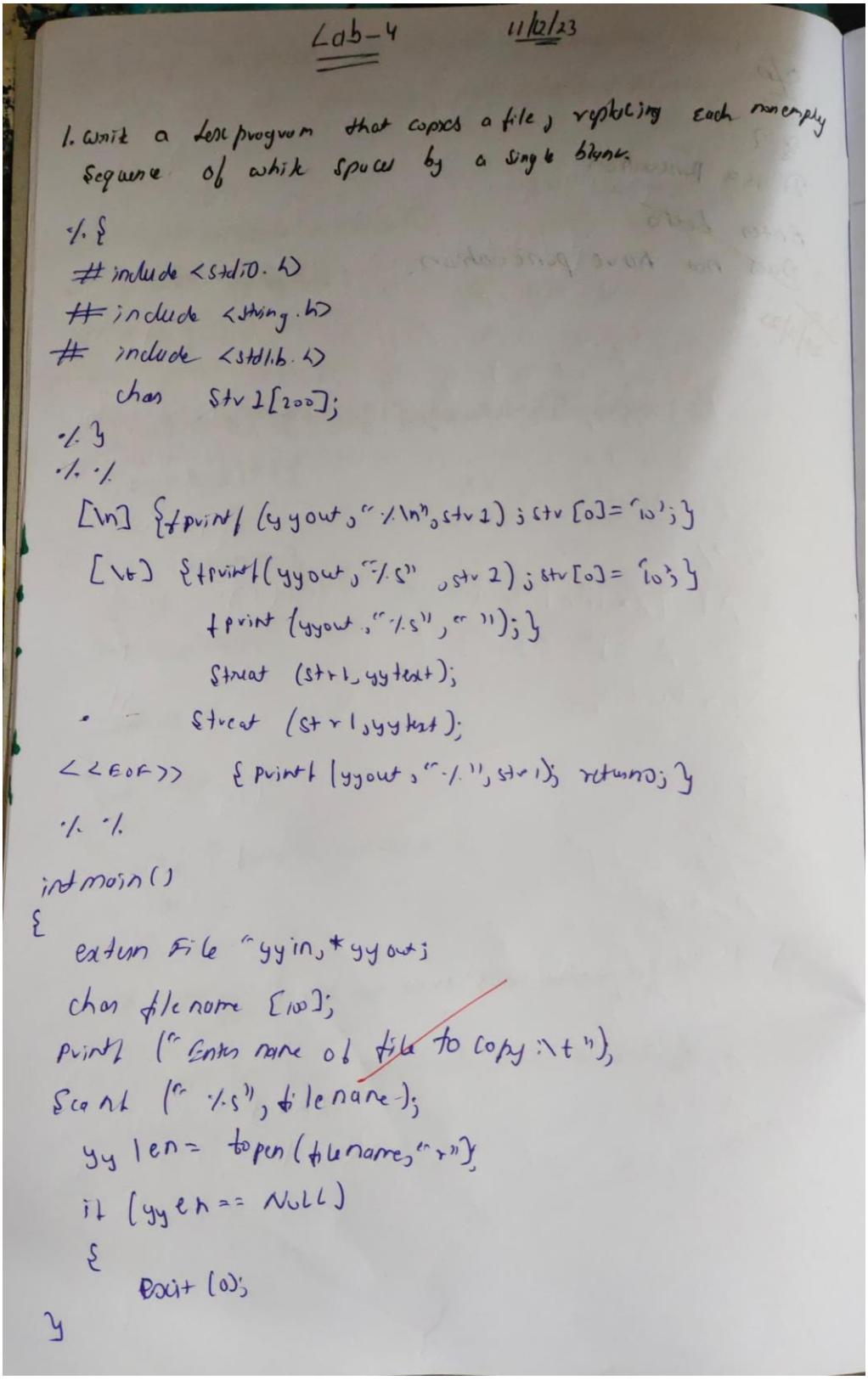
Ans O/P

```
lex jigs.l
cc lex.yy.c
./a.out
Enter num: +8
Its a numbers: +8

+7.9
Its a float point num: +7.9
```

Output:


bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ lex float.l
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ gcc lex.yy.c
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
enter any number 23.6
floating point numbers
45
not a floating point number
+6.3
floating point numbers
-55.66
floating point numbers
55.
not a floating point number

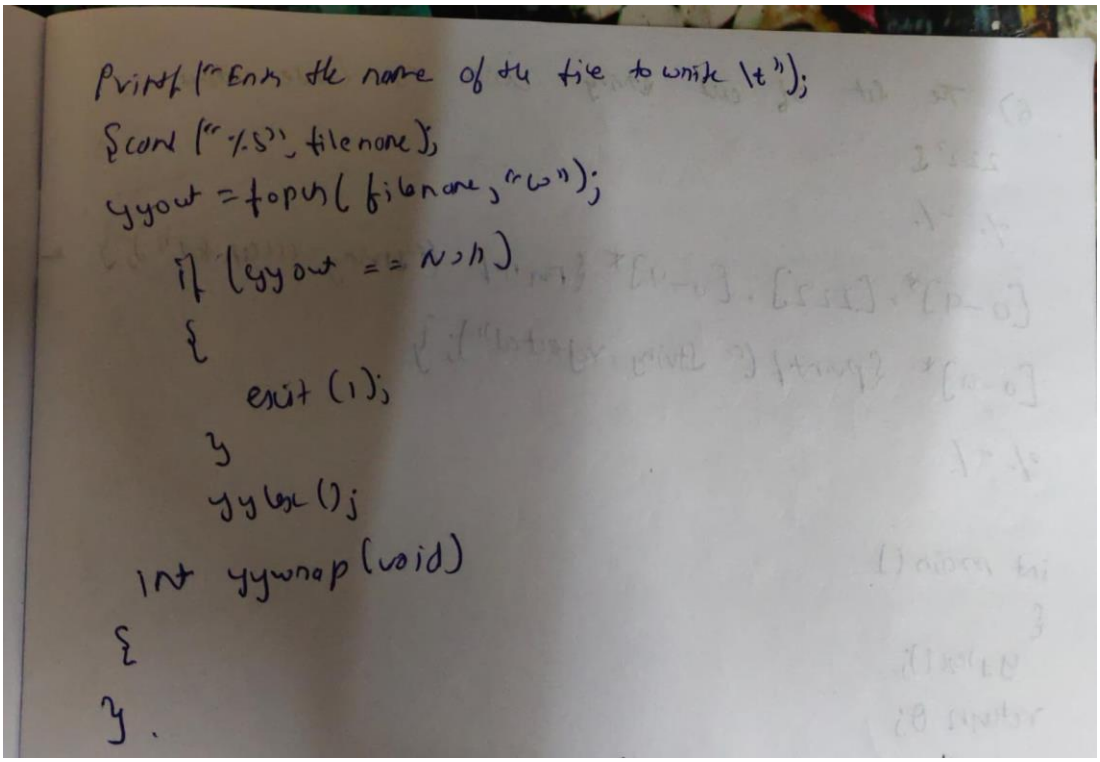
Code:



Lab-4 11/10/23

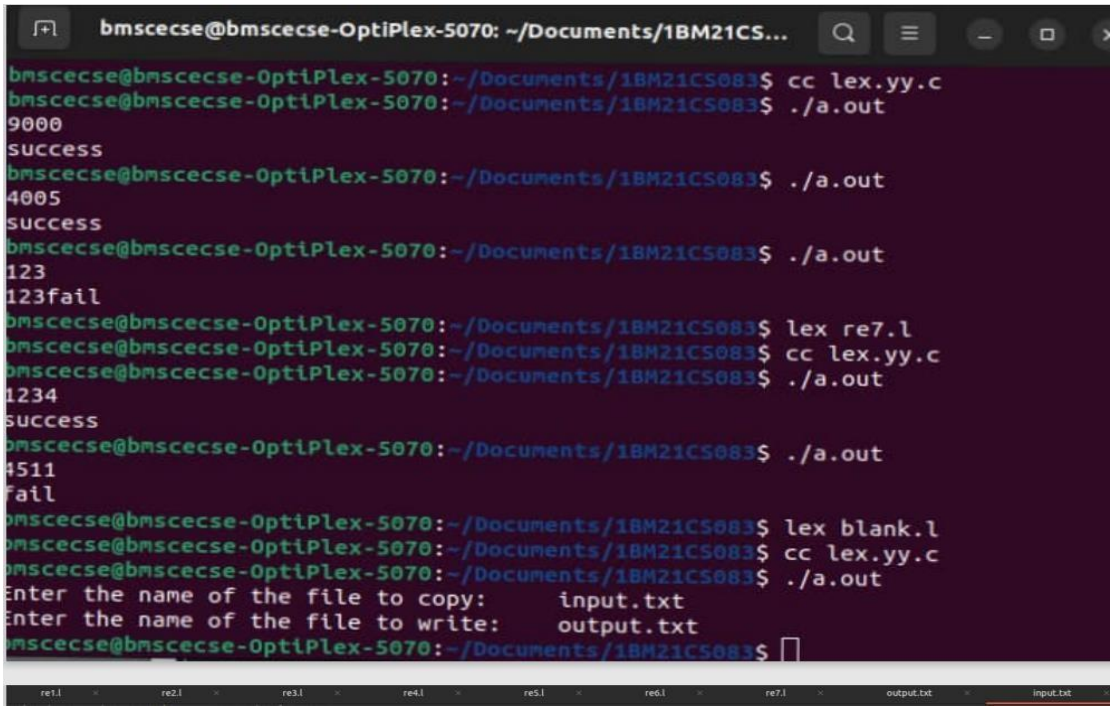
1. Write a Lex program that copies a file, replacing each nonempty sequence of white spaces by a single blank.

```
%{
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
    char str1[200];
%}
%%
[\n] {fprintf(yyout,"%s\n",str1); str[0]='\0';}
[\t] {fprintf(yyout,"%s",str1); str[0]='\0';}
        fprintf(yyout,"%s",""));}
        Strcat(str1,yytext);
    .   Strcat(str1,yytext);
<<EOF>>  {printf(yyout,"%s",str1); return 0;}
%%
int main()
{
  extern FILE *yyin,*yyout;
  char filename[100];
  printf("Enter name of file to copy:\t");
  Scanf("%s",filename);
  yyin = fopen(filename,"r");
  if(yyin == NULL)
  {
     exit(0);
}
```



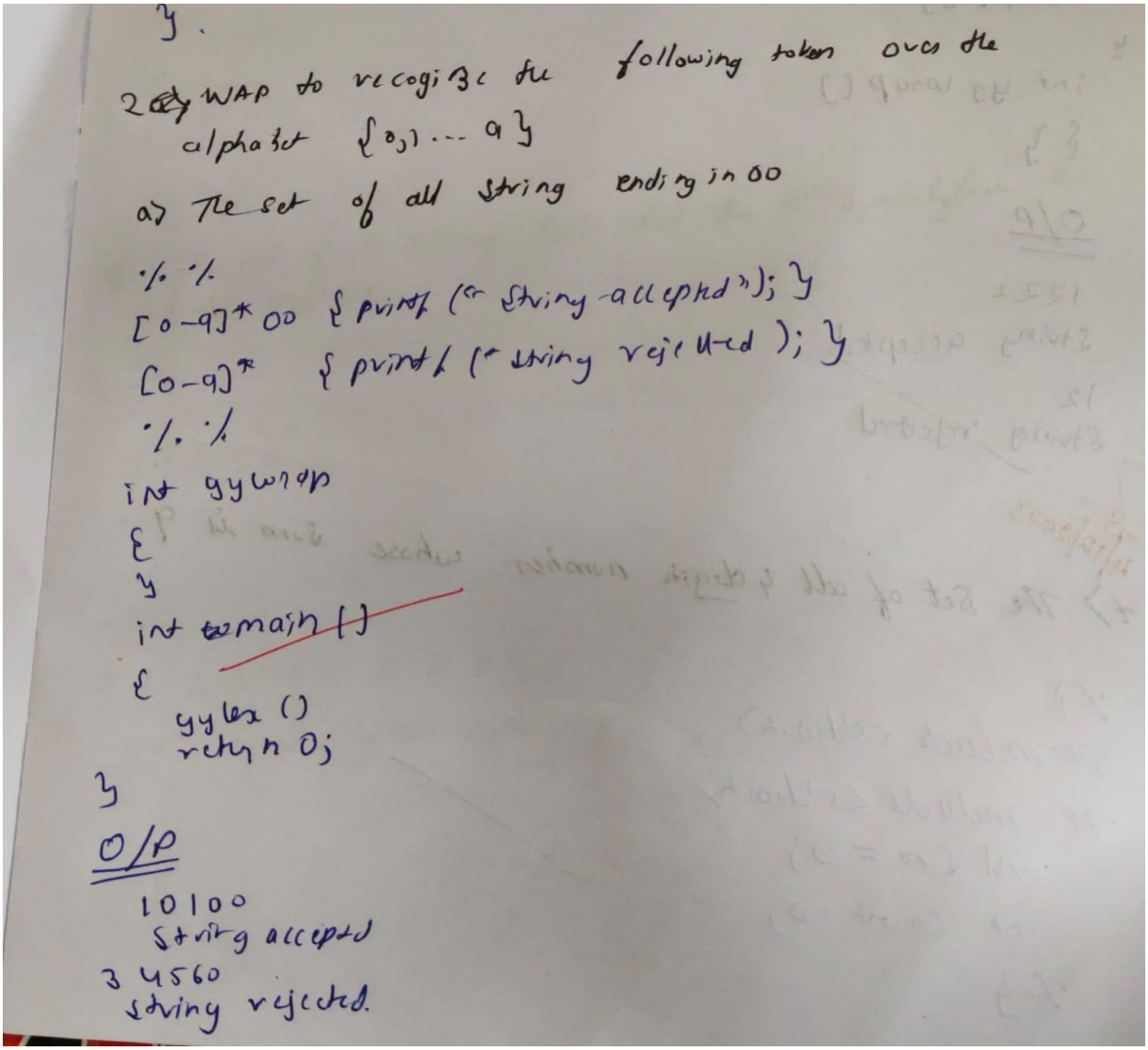
```
Printf("Enter the name of the file to write\t");
fscanf("%s", filename);
yyout = fopen(filename, "w");
  if (yyout == NULL)
  {
    exit(1);
  }
  yylex();
int yywrap(void)
{
}.
```

Output:



```
bmscecse@bmscecse-OptiPlex-5070: ~/Documents/1BM21CS...
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ cc lex.yy.c
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
9000
success
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
4005
success
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
123
123fail
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ lex re7.l
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ cc lex.yy.c
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
1234
success
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
4511
fail
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ lex blank.l
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ cc lex.yy.c
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
Enter the name of the file to copy:     input.txt
Enter the name of the file to write:    output.txt
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$
re1.l  re2.l  re3.l  re4.l  re5.l  re6.l  re7.l  output.txt  input.txt
```

Code:



```
}.

2) WAP to recognize the following token over the
   alphabet {0,1 ... 9}
a) The set of all string ending in 00

%%
[0-9]*00 { printf("string accepted"); }
[0-9]*   { printf("string rejected"); }
%%
int yywrap
{
}
int main()
{
   yylex()
   return 0;
}
```

O/P

```
10100
String accepted
34560
string rejected.
```

b) The set of all string with 3 - consecutive

```
222'$
%%
[0-9]*.[222].[0-9]* {printf ("string accepted"); }
[0-9]* {printf ("string rejected"); }
%%

int main()
{
  yylex();
return 0;
}
  int yywrap()
  {}
```

O/P

1222
String accepted
12
String rejected

4/12/2023

4) The set of all 4 digits numbers whose sum is 9

```
%{
#include <stdio.h>
#  include <stdio.h>
   int sum = 0;
   int count = 0;
%}
```

```
%%
[0-9] { sum = sum + atoi (yytext); count++; }
\n { if sum % y == 0 && count == 4 }
    { printf ("yes\n"); sum = 0; count = 0; }
    { }
%%
int yywrap()
{
}
int main()
{
  yylex();
  return 0;
}
```

O/P

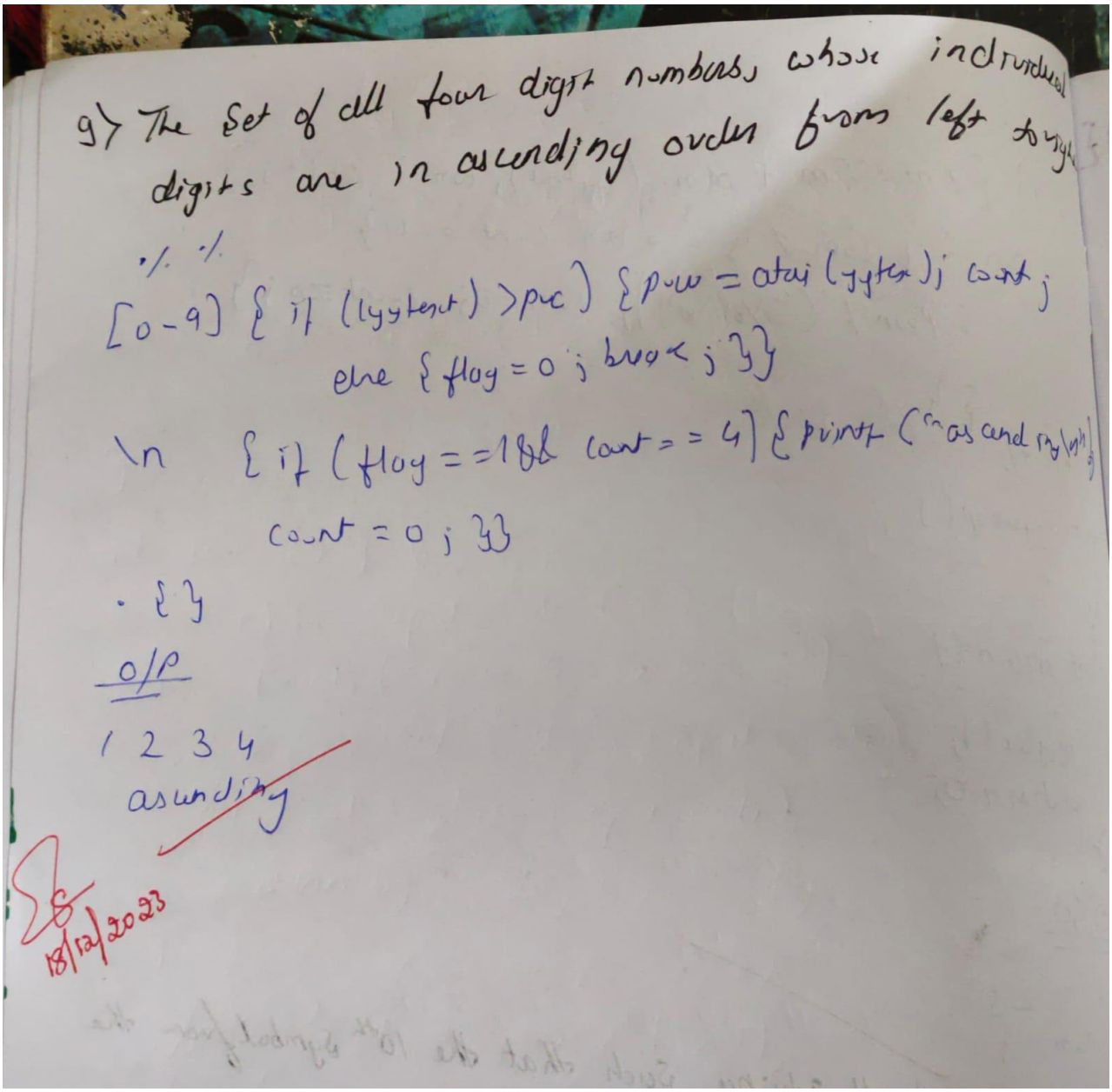
1 2 2 3
sum = 9

(8) The set of all string such that the 10th symbol from the right end is 1.

```
%%
([1])*1{1}{9}{printf("string accepted");}
%%
```

O/P

0 1 2 3 4 5 6 7 8 9 1
10th symbol from left is 9



9) The set of all four digit numbers whose individual digits are in ascending order from left to right

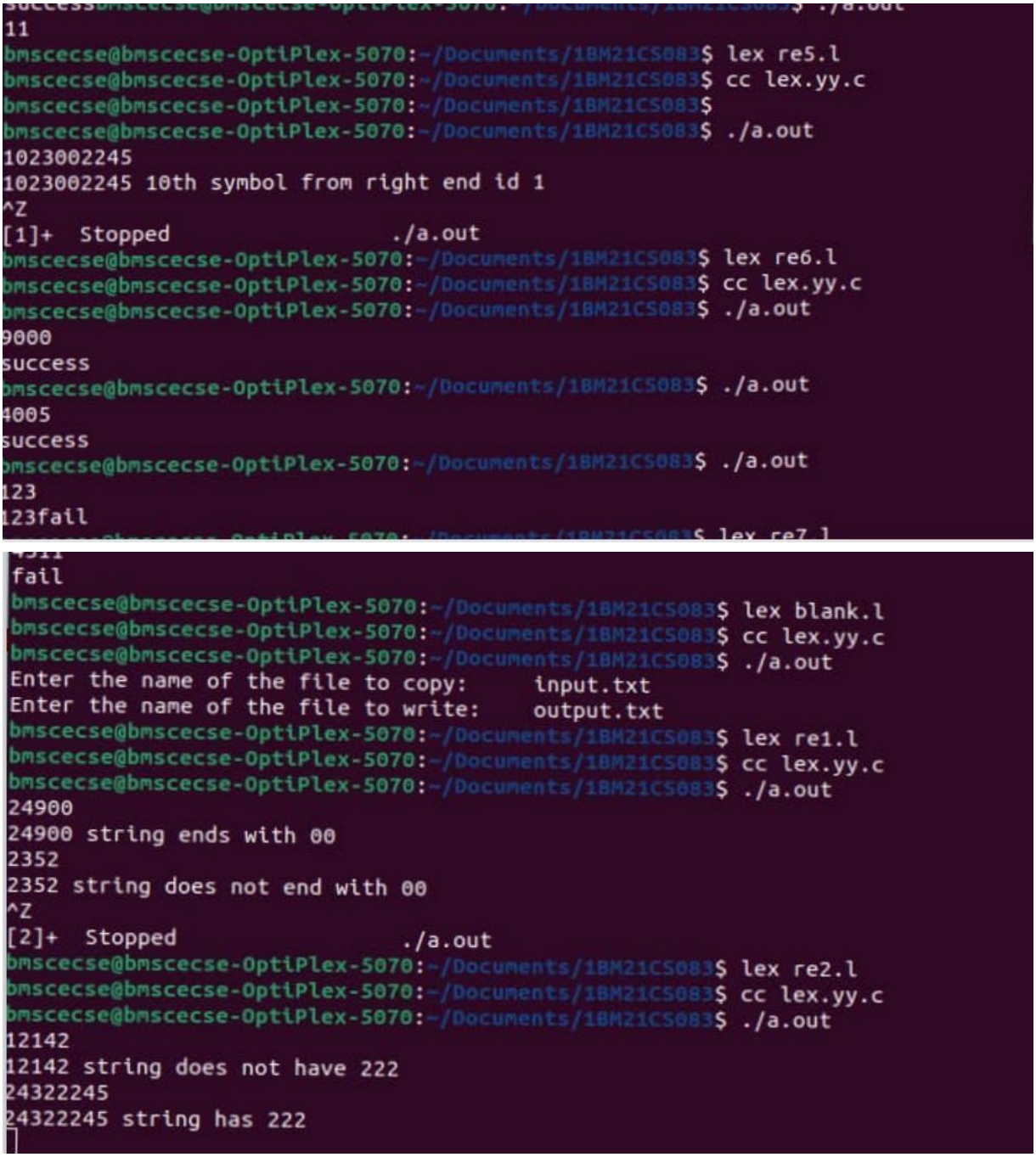
```
%%
[0-9] { if (yytext[0] > prc) { prc = atoi(yytext); count++; }
        else { flag = 0; break; } }
\n      { if (flag == 1 && count == 4) { printf("\nascending order"); }
          count = 0; }
.{}
```

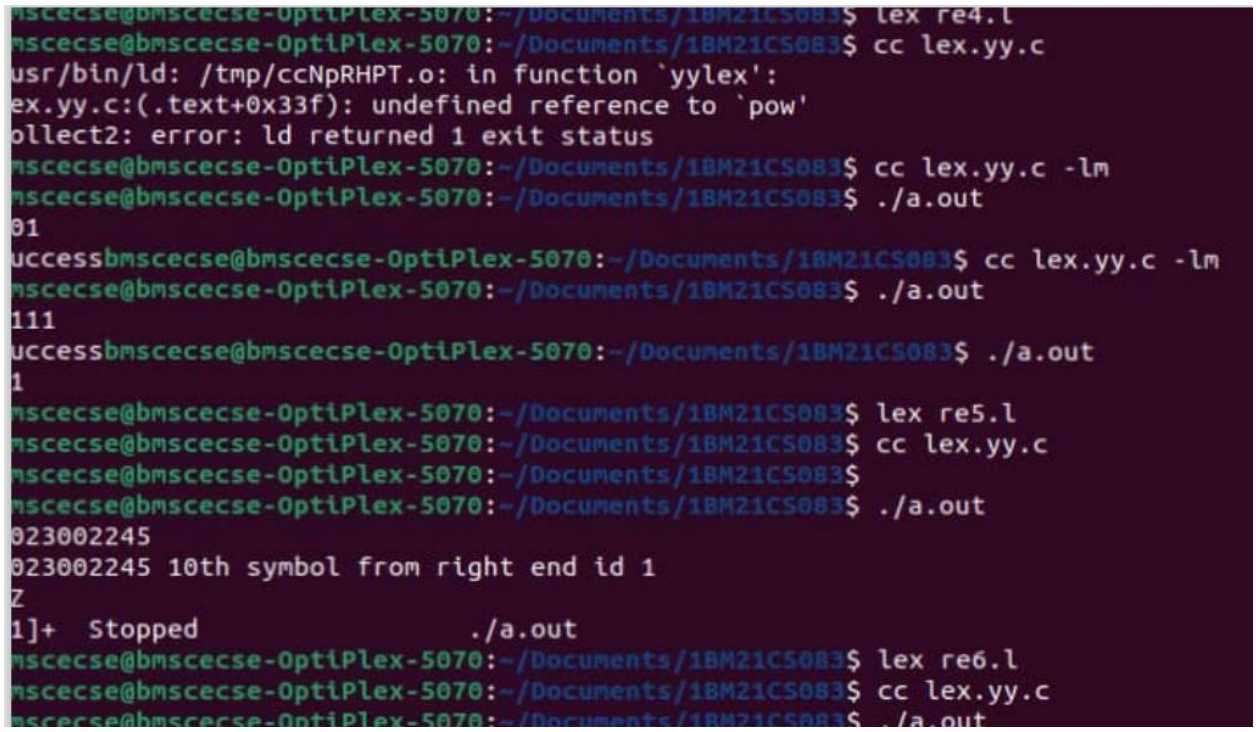
o/p

1 2 3 4
ascending

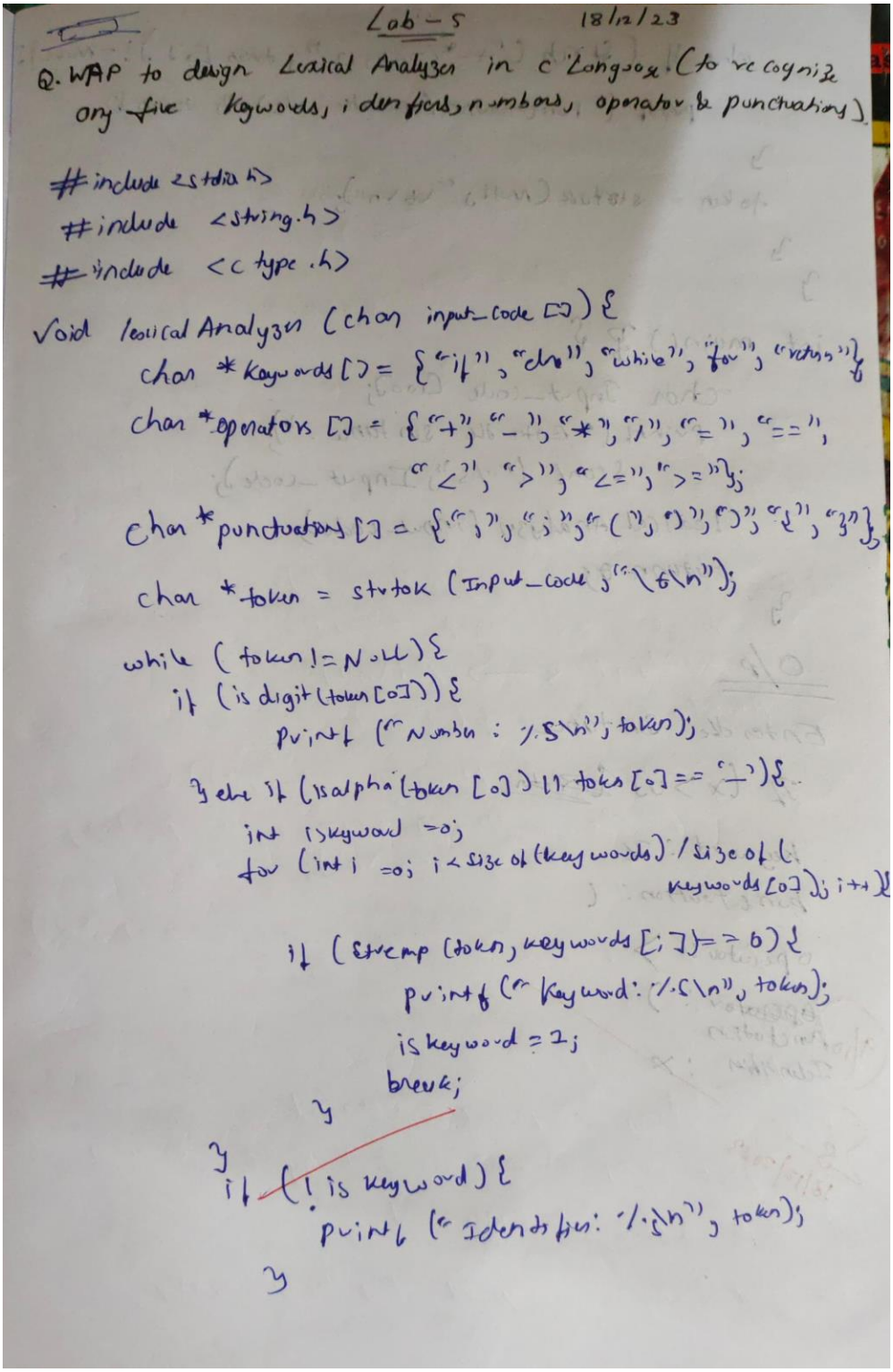
18/12/2023

Output:


11
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ lex re5.l
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ cc lex.yy.c
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
1023002245
1023002245 10th symbol from right end id 1
^Z
[1]+ Stopped ./a.out
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ lex re6.l
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ cc lex.yy.c
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
9000
success
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
4005
success
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
123
123fail
fail
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ lex blank.l
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ cc lex.yy.c
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
Enter the name of the file to copy: input.txt
Enter the name of the file to write: output.txt
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ lex re1.l
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ cc lex.yy.c
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
24900
24900 string ends with 00
2352
2352 string does not end with 00
^Z
[2]+ Stopped ./a.out
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ lex re2.l
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ cc lex.yy.c
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
12142
12142 string does not have 222
24322245
24322245 string has 222


mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ lex re4.l
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ cc lex.yy.c
usr/bin/ld: /tmp/ccNpRHPT.o: in function `yylex':
ex.yy.c:(.text+0x33f): undefined reference to `pow'
ollect2: error: ld returned 1 exit status
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ cc lex.yy.c -lm
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
01
uccessbmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ cc lex.yy.c -lm
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
111
uccessbmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
1
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ lex re5.l
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ cc lex.yy.c
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
023002245
023002245 10th symbol from right end id 1
Z
1]+ Stopped ./a.out
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ lex re6.l
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ cc lex.yy.c

code:



Lab - 5 18/12/23

Q. WAP to design Lexical Analyzer in c Language. (to recognize any five keywords, identifiers, numbers, operator & punctuations).

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

void lexicalAnalyzer (char input_code []) {
    char *keywords [] = {"if", "else", "while", "for", "return"};
    char *operators [] = {"+", "-", "*", "/", "=", "==",
                          "<", ">", "<=", ">="};
    char *punctuations [] = {",", ";", "(", ")", "{", "}"};
    char *token = strtok (input_code, "\t\n");

    while (token != NULL) {
        if (isdigit(token[0])) {
            printf ("Number : %s\n", token);
        } else if (isalpha(token[0]) || token[0] == '_') {
            int iskeyword = 0;
            for (int i = 0; i < sizeof(keywords) / sizeof(
                                keywords[0]); i++) {

                if (strcmp (token, keywords[i]) == 0) {
                    printf ("Keyword: %s\n", token);
                    iskeyword = 1;
                    break;
                }
            }
            if (!iskeyword) {
                printf ("Identifier: %s\n", token);
            }
```

```
    else if (strchr("+-*/=<>;(){}", token[0]) != NULL)
        printf("Punctuation/operator: %s\n", token);
    }
    token = strtok(NULL, " \t\n");
  }
}

int main() {
    char input_code[100];
    printf("Enter the sentence:");
    scanf("%s", input_code);
    lexical Analysis(input_code);
    return 0;
}
```

O/P

Enter the sentence

If (x > 0) ~~Syntax~~

Keyword : If

punctuation : (

operator : >

~~Operator~~ :)

Punctution

Identifier : x

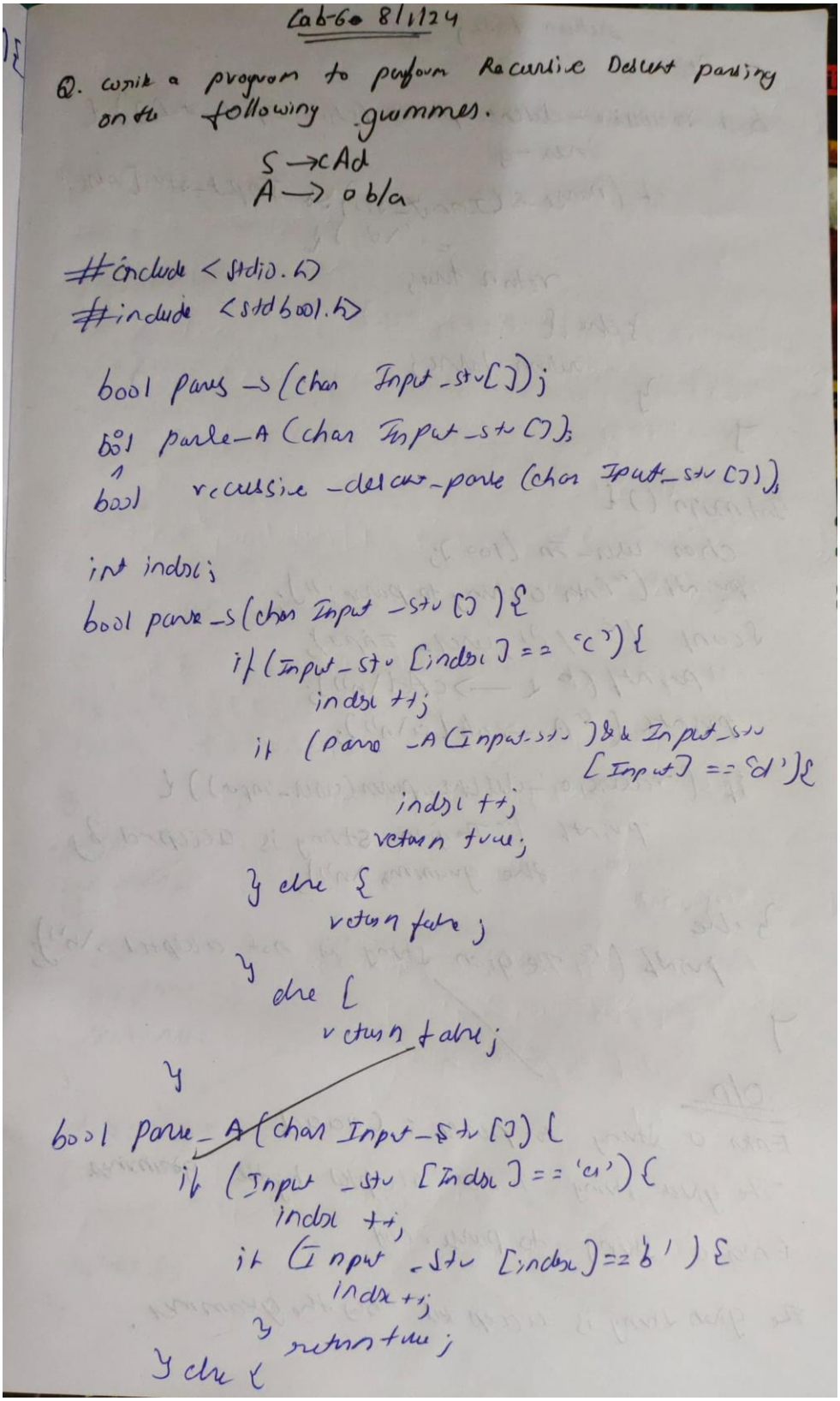
9/10

18/12/2023

Output:

```
enter c code
int a = 1234 ;
Keyword: int
Identifier: a
Punctuation/Operator: =
Number: 1234
Punctuation/Operator: ;
```

Code:



Lab-6 8/1/24

Q. write a program to perform Recursive Descent parsing on the following grammer.

S → cAd
A → ob/a

```
#include <stdio.h>
#include <stdbool.h>

bool parse_S(char Input_str[]);
bool parse_A(char Input_str[]);
bool recursive_descent_parse(char Input_str[]);

int index;
bool parse_S(char Input_str[]) {
    if (Input_str[index] == 'c') {
        index++;
        if (parse_A(Input_str) && Input_str
                [Input] == 'd') {
            index++;
            return true;
        } else {
            return false;
        }
    } else {
        return false;
    }
}

bool parse_A(char Input_str[]) {
    if (Input_str[index] == 'a') {
        index++;
        if (Input_str[index] == 'b') {
            index++;
        } return true;
    } else {
```

```
    return false;
}
bool recursive_descent_parser(char input_str[]) {
    indx = 0;
    if (parse_S(input_str) && input_str[indx]
        == '\0') {
        return true;
    } else {
        return false;
    }
}
int main() {
    char user_input[100];
    printf("Enter a string to parse: ");
    scanf("%s", user_input);
    printf("S -> cAd\n");
    printf("A -> ab | a\n");
    if (recursive_descent_parser(user_input)) {
        printf("The given string is accepted by
            the grammar\n");
    } else
        printf("The given string is not accepted.\n");
}
```

O/P

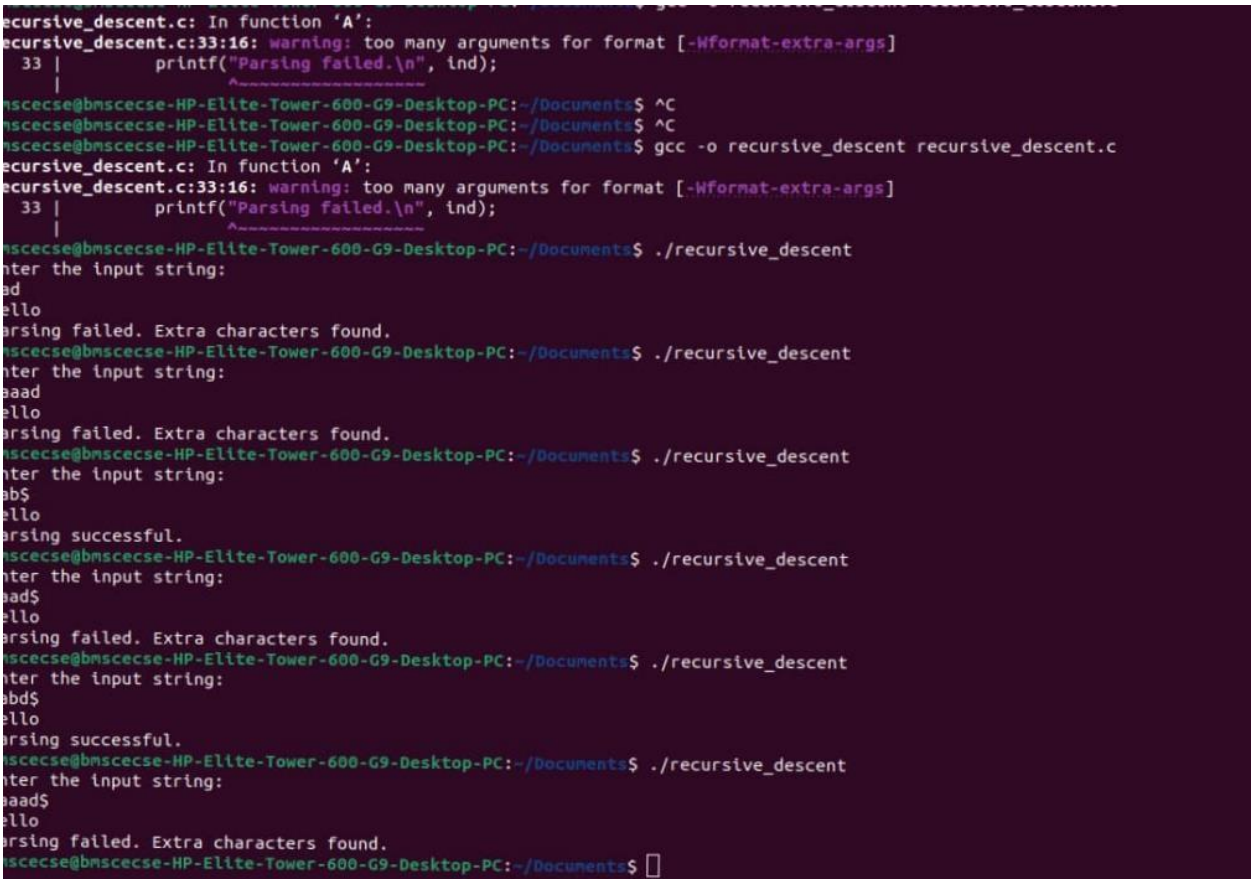
Enter a string to parse : caaad

The given string is not accepted by the grammar

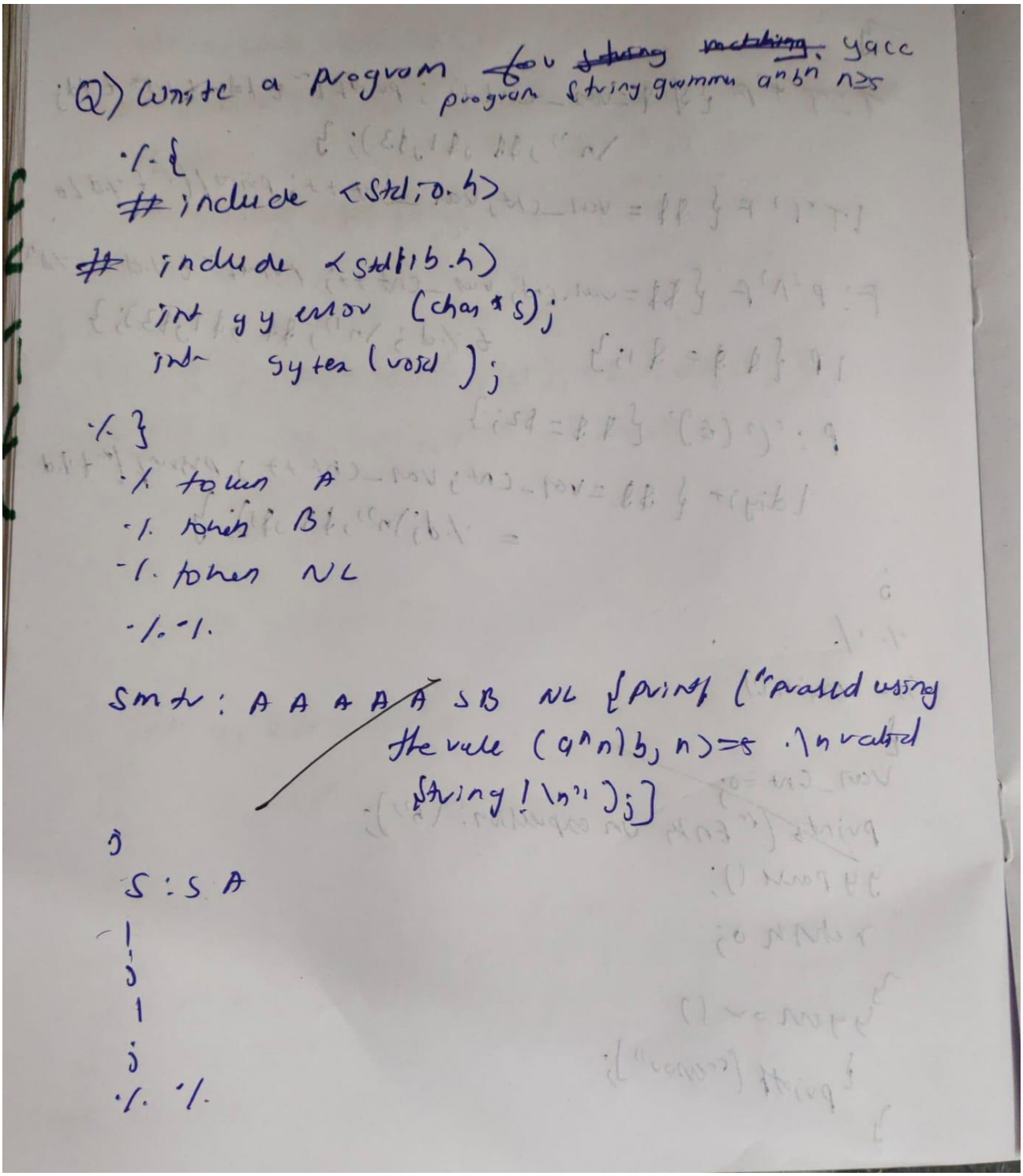
Enter a string to parse : cad

The given string is accepted by the grammar.

Output:

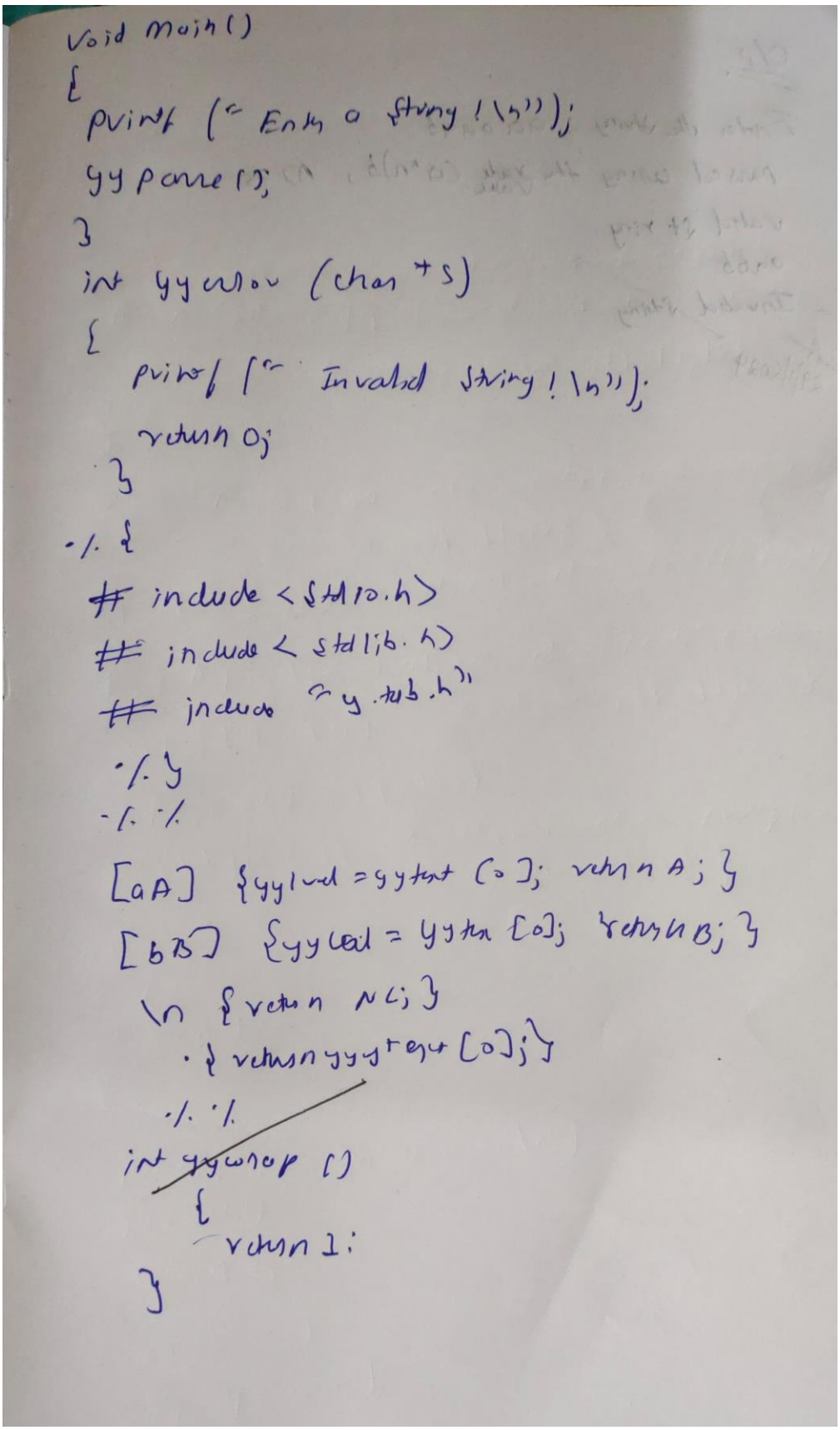

ecursive_descent.c: In function 'A':
ecursive_descent.c:33:16: warning: too many arguments for format [-Wformat-extra-args]
33 | printf("Parsing failed.\n", ind);
| ^~~~~~~~~~~~~~~~~~~
nscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ ^C
nscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ ^C
nscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ gcc -o recursive_descent recursive_descent.c
ecursive_descent.c: In function 'A':
ecursive_descent.c:33:16: warning: too many arguments for format [-Wformat-extra-args]
33 | printf("Parsing failed.\n", ind);
| ^~~~~~~~~~~~~~~~~~~
nscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ ./recursive_descent
nter the input string:
ad
ello
arsing failed. Extra characters found.
nscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ ./recursive_descent
nter the input string:
aaad
ello
arsing failed. Extra characters found.
nscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ ./recursive_descent
nter the input string:
ab$
ello
arsing successful.
nscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ ./recursive_descent
nter the input string:
aad$
ello
arsing failed. Extra characters found.
nscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ ./recursive_descent
nter the input string:
abd$
ello
arsing successful.
nscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ ./recursive_descent
nter the input string:
aaad$
ello
arsing failed. Extra characters found.
nscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$

Code:



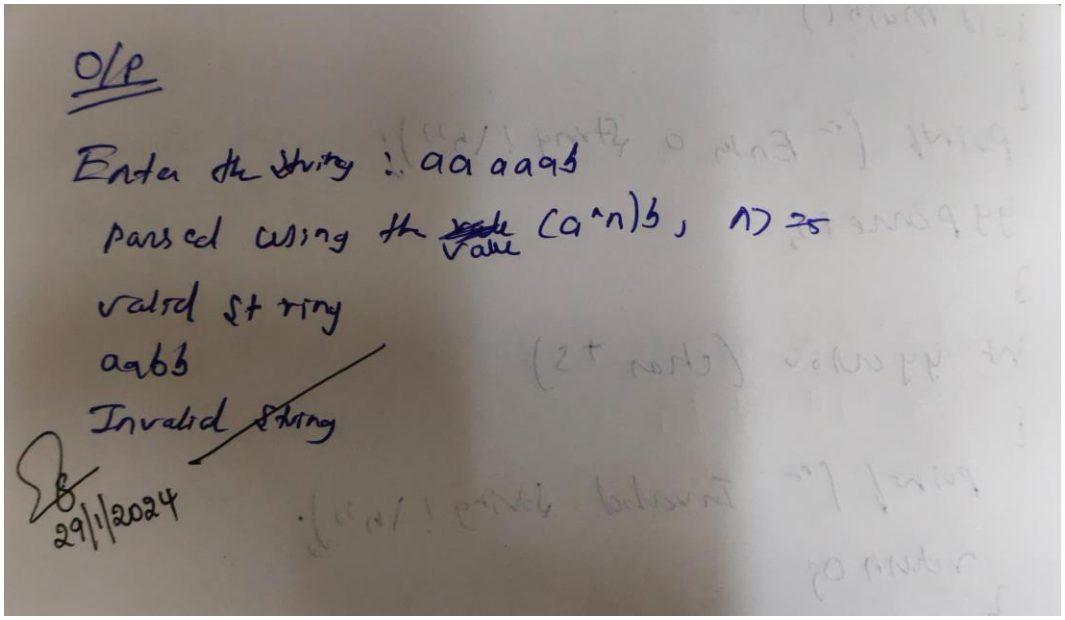
Q) Write a program ~~for~~ ~~lexing~~ ~~matching~~ yacc program string grammar $a^n b^n$ $n \geq 5$

```
%{
#include <stdio.h>
#include <stdlib.h>
int yyerror(char *s);
int yylex(void);
%}
%token A
%token B
%token NL
%%
Start: A A A A A S B NL { printf("Valid using the rule (a^n)b, n>=5 \n Invalid string !\n"); }
;
S: S A
|
;
%%
```



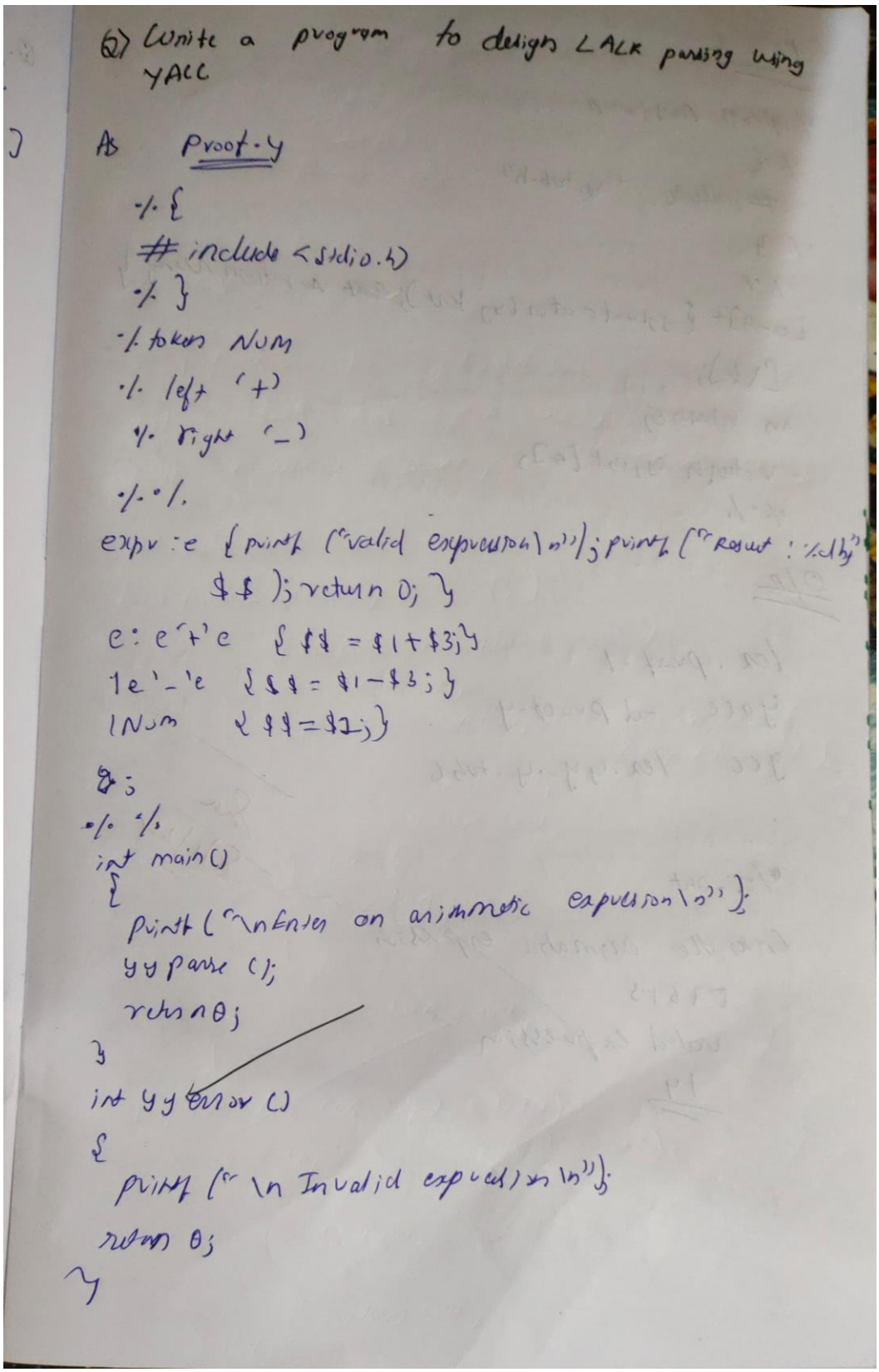
```
void main()
{
printf ("Enter a string!\n");
yyparse();
}
int yywrap (char *s)
{
   printf ("Invalid string!\n");
   return 0;
  }
%{
#include <stdio.h>
#include <stdlib.h>
#include "y.tab.h"
%}
%%
[aA] {yylval = yytext[0]; return A;}
[bB] {yylval = yytext[0]; return B;}
\n {return NL;}
. {return yytext[0];}
%%
int yywrap()
   {
    return 1;
}
```

Output:




```
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ lex anbn.l
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ yacc -d anbn.y
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ gcc lex.yy.c y.tab.c
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
nter a string!
abb$
nvalid String!
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
nter a string!
abb
nvalid String!
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
nter a string!
aaab
nvalid String!
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
nter a string!
aaaab
arsed using the rule (a^n)b, n>=5.
alid String!
aaaaaabb
nvalid String!
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$
```

Code:



2) Write a program to design LALR parsing using YACC

As Proof.y

```
%{
#include<stdio.h>
%}
%token Num
%left '+'
%right '-'
%%
expr:e {printf("valid expression\n");printf("Result : %d\n",
        $$);return 0;}
e:e'+'e  {$$=$1+$3;}
|e'-'e  {$$=$1-$3;}
|Num    {$$=$1;}
;
%%
int main()
{
  printf("\nEnter an arithmetic expression\n");
  yyparse();
  return 0;
}
int yyerror()
{
  printf("\n Invalid expression\n");
  return 0;
}
```

proof.l

```
%option noyywrap
%{
#include "y.tab.h"
%}
%%
[0-9]+ {yylval = atoi(yytext); return NUM;}
[\t];
\n return 0;
. return yytext[0];
%%
```

O/P

```
lex proof.l
yacc -d proof.y
gcc lex.yy.c y.tab.c
./a.out
```

Enter the arithmetic expression

5+6+3

Valid expression

11

8/1/24

```
bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ lex proo1.l
bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ yacc -d proo1.y
bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ gcc lex.yy.c y.tab.c
y.tab.c: In function 'yyparse':
y.tab.c:1022:16: warning: implicit declaration of function 'yylex' [-Wimplicit-function-declaration]
 1022 |       yychar = yylex ();
      |                ^~~~~
y.tab.c:1205:7: warning: implicit declaration of function 'yyerror'; did you mean 'yyerrok'? [-Wimplicit-function-declaration]
 1205 |       yyerror (YY_("syntax error"));
      |       ^~~~~~~
      |       yyerrok
bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ ./a.out

Enter an arithmetic expression
5+6
Valid expression
Result : 11
bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ ./a.out

Enter an arithmetic expression
5*6-2
Valid expression
Result : 28
bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ ./a.out

Enter an arithmetic expression
5-6+*

Invalid expression
bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$
```

29/10/24 Lab - 7

* Write a Yacc program to generate syntax trees for a given arithmetic expression.

P1.l

```
%{
#include "y.tab.h"
extern int yylval;
%}
%%
[0-9]+ { yylval = atoi(yytext); return digit; }
[\t];
[\n] return 0;
. return yytext[0];
%%
int yywrap()
{
}
```

P1.y

```
%{
#include <math.h>
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct tree_node
{
    char val[10];
    int l;
    int r;
};
int ind;
```

```
struct tree_node syn_tree[100];
void my_print_tree(int curr_ind);
int mknode(int lc, int rc, char val[10]);
%}
%token digit
%%
S: E {my_print_tree($1);}
;
E: E'+' T {$$ = mknode($1, $3, "+");}
| T {$$ = $1;}
;
T: T'*' F {$$ = mknode($1, $3, "*");}
| F {$$ = $1;}
;
F: '('E')' {$$ = $2;}
| digit {char buf[10]; sprintf(buf, "%d", yylval); $$ = mknode(-1, -1, buf);}
;
%%
int main()
{
  ind=0;
printf("Enter an expression\n");
yyparse();
return 0;
}
int yyerror()
{
  printf("NITW Error\n");
}
int mknode(int lc, int rc, char val[10])
{
```

```
strcpy(syn_tree[ind].val,val);
syn_tree[ind].lc = lc;
syn_tree[ind].rc = rc;
ind++;
return ind-1;
}
void my_printtree(int cur_ind)
{
if(cur_ind == -1) return;
if(syn_tree[cur_ind].lc == -1 && syn_tree[cur_ind].
                                rc == -1)
printf("Digit Node -> Index : %d, value : %s \n", cur_
                    ind, syn_tree[cur_ind].val);
my_printtree(syn_tree[cur_ind].lc);
my_printtree(syn_tree[cur_ind].rc);
}
```

O/P

Command

```
lex p1.l
yacc p1.y
gcc lex.yy.c y.tab.c
./a.out
```

Enter an expression

2+3*5

Operator Node -> Index : 4, value : +, left child Index : 0,
Right child Index : 3

Leaf Node -> Index 0, value 2

operator Node -> Index : 3, value : *, left child Index : 1,
Right child Index : 2

Leaf Node -> Index : 1, value : 3

Leaf Node -> Index : 2, value : 5

```
bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ ./a.out
Enter an expression
4+6*9
Operator Node -> Index : 4, Value : +, Left Child Index : 0,Right Child Index : 3
Digit Node -> Index : 0, Value : 4
Operator Node -> Index : 3, Value : *, Left Child Index : 1,Right Child Index : 2
Digit Node -> Index : 1, Value : 6
Digit Node -> Index : 2, Value : 9
bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$
```

Q. Use YACC to convert: Infix expression to Postfix expression.

P4.l

```
%{
#include "y.tab.h"
extern int yylval;
%}
%%
int yywrap()
{
}
```

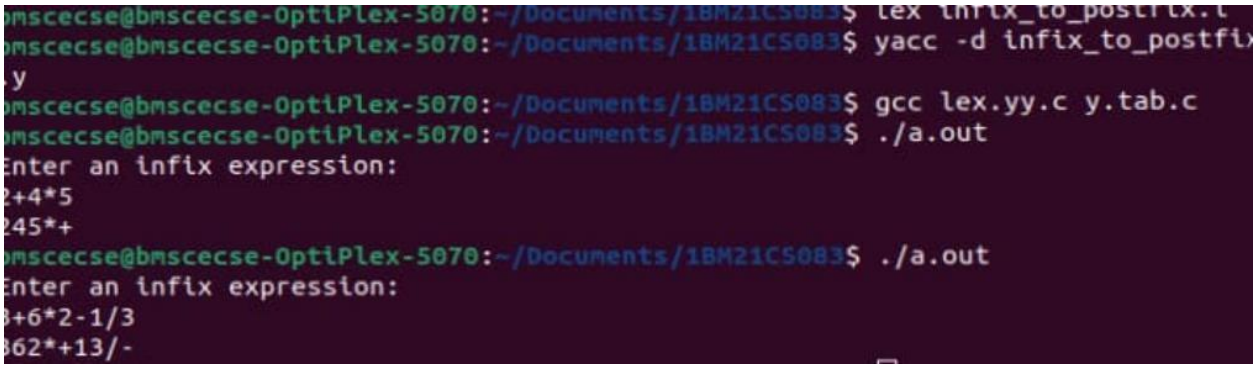
P4.y

```
%{
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
%}
%token digit
%%
S : E { printf("\n\n"); }
;
E : E '+' T { printf("+"); }
| T
T : T '*' F { printf("*"); }
| F
F : '(' E ')'
| digit { printf("%d", $1); }
;
%%
```

```
int main()
{
printf("Enter Infix expression : ");
yyparse();
}
yyerror()
{
  printf("Error");
}
```

Output

```
lex p4.l
yacc p4.y
gcc lex.yy.c y.tab.c
./a.out
Enter Infix expression : 2+6*3+4
2 6 3 * + 4 +
```


mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ lex infix_to_postfix.l
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ yacc -d infix_to_postfix
.y
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ gcc lex.yy.c y.tab.c
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
Enter an infix expression:
2+4*5
245*+
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
Enter an infix expression:
3+6*2-1/3
362*+13/-

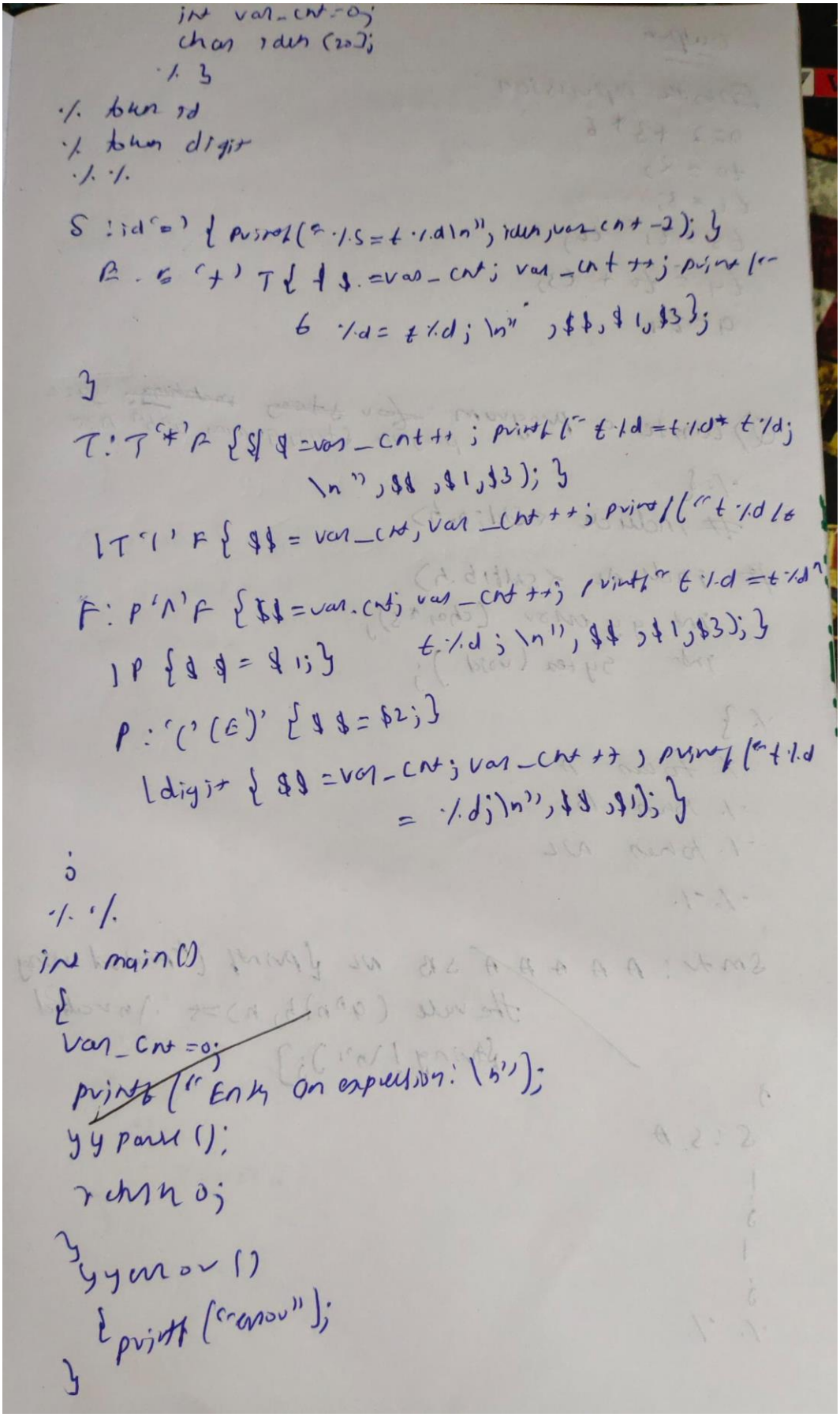
Q. use YACC to generate 3- address code for a given expression.

P1.l

```
d[0-9]+
a[a-z,A-Z]+
%{
#include <stdio.h>
#include <stdlib.h>
#include "y.tab.h"
extern int yylval;
extern char idn[20];
%}
%%
{d} { yylval = atoi(yytext); return digit; }
{a} { strcpy(idn, yytext); yylval
[\t] {;}
\n return 0;
. return yytext[0];
%%
int yywrap()
{
}
```

P1.y

```
%{
#include <math.h>
#include <ctype.h>
#include <stdio.h>
```



```
int var_cnt=0;
char idstr[20];
%}
%token id
%token digit
%%
S : id '=' { printf("%d\t%s\t=\t%d\n", idstr, var_cnt-2); }
  | E
E : E '+' T { $$=var_cnt; var_cnt++; printf("%d\t=\t%d\t+\t%d\n", $$, $1, $3); }
  | T
T : T '*' F { $$=var_cnt++; printf("%d\t=\t%d\t*\t%d\n", $$, $1, $3); }
  | F
F : P '^' F { $$=var_cnt; var_cnt++; printf("%d\t=\t%d\t^\t%d\n", $$, $1, $3); }
  | P { $$=$1; }
P : '(' E ')' { $$=$2; }
  | digit { $$=var_cnt; var_cnt++; printf("%d\t=\t%d\n", $$, $1); }
;
%%
int main()
{
var_cnt=0;
printf("Enter an expression:\n");
yyparse();
return 0;
}
yyerror()
{printf("error");
}
```

```
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ lex 3addcode.l
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ yacc -d 3addcode.y
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ gcc lex.yy.c y.tab.c
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
nter an expression:
=8+9-2
0 = 8;
1 = 9;
2 = t0 + t1;
3 = 2;
4 = t2 - t3;
=t4
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
nter an expression:
=2^3/23+5
 = 2;
1 = 3;
2 = t0 ^ t1;
 = 23;
4 = t2 / t3;
 = 5;
 = t4 + t5;
t6
```