VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT

on

COMPILER DESIGN

Submitted by

Girish Kumar S K(1BM21CS068)

Under the Guidance of
Prameetha Pai
AssistantProfessor,
BMSCE

in partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING

(Autonomous Institution under VTU)

BENGALURU-560019

November 2023-February 2024

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 **Girish Kumar S K (1BM21CS068)** , who is bonafide student of **B. M. S. College of Engineering.** It is in partial fulfilment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2023-24.

The Lab report has been approved as it satisfies the academic requirements in respect of **Compiler Design- (22CS5PCCPD)** work prescribed for the said degree.

Prameetha Pai	Dr. Jyothi Nayak
Assistant professor	Professor and Head
Department of CSE	Department of CSE
BMSCE, Bengaluru	BMSCE, Bengaluru

B. M. S. COLLEGE OF ENGINEERING

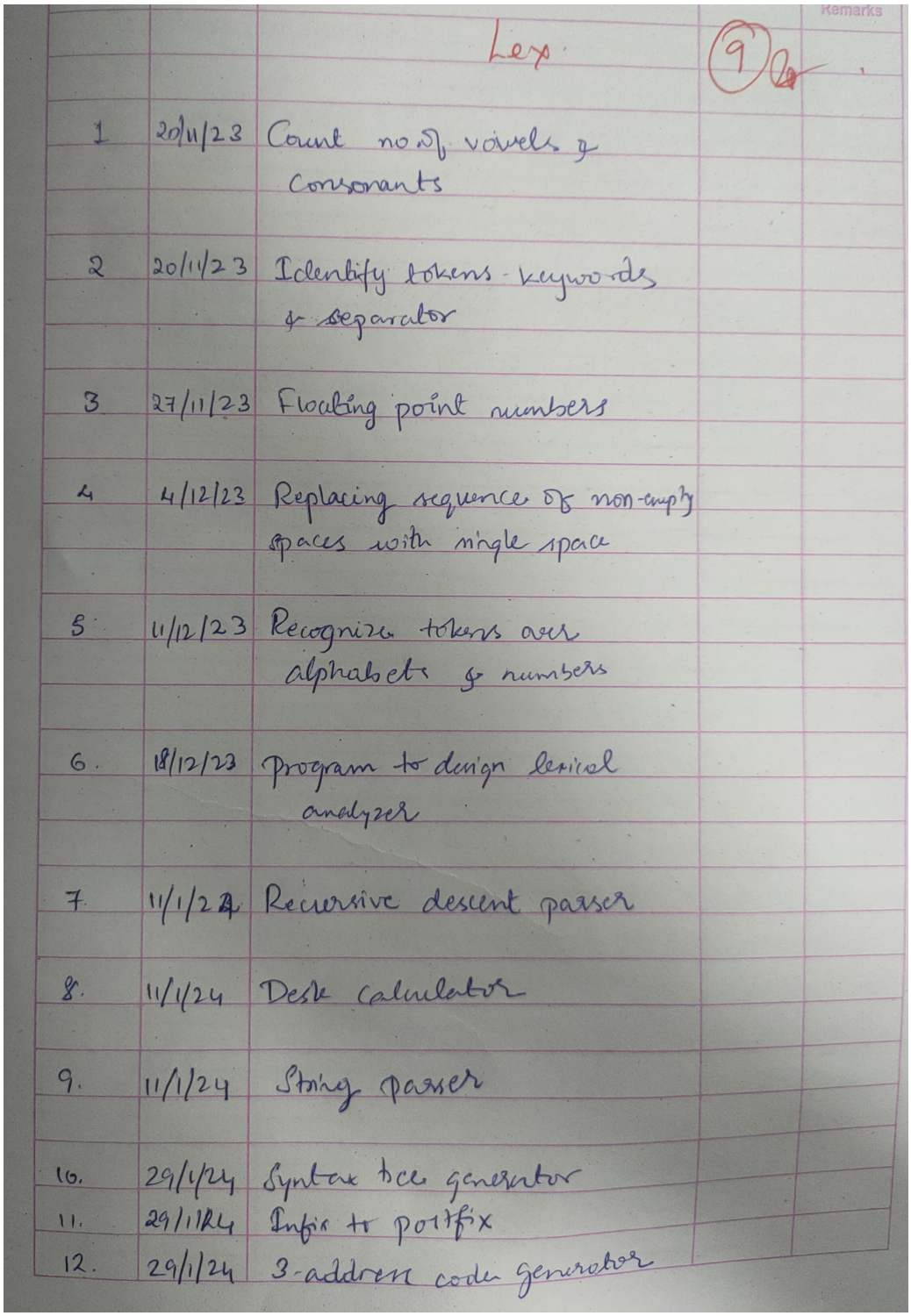
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

I, Girish Kumar S K (1BM21CS068), student of 5th Semester, B.E, Department of Computer Science and Engineering, B. M. S. College of Engineering, Bangalore, here by declare that, this lab report entitled " **Compiler Design**" has been carried out by me under the guidance of Prameetha Pai, Assistant Professor, Department of CSE, B. M. S. College of Engineering, Bangalore during the academic semester November-2023-February-2024.

I also declare that to the best of my knowledge and belief, the development reported here is not from part of any other report by any other students.

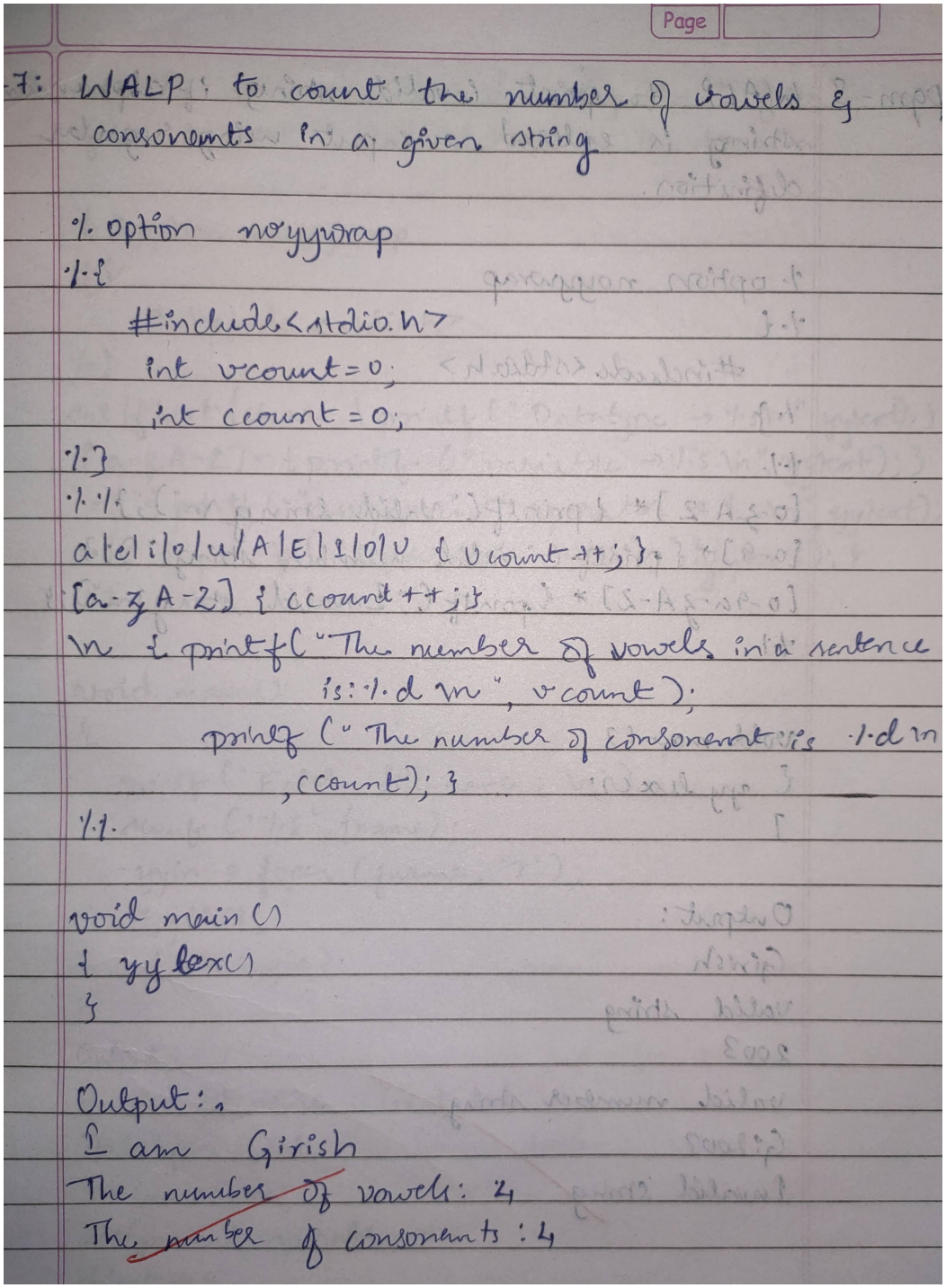
TABLE OF CONTENTS



Lex

Remarks

1	20/11/23	Count no of vowels & consonants	
2	20/11/23	Identify tokens - keywords & separator	
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 & numbers	
6.	18/12/23	Program to design lexical analyzer	
7.	11/1/24	Recursive descent parser	
8.	11/1/24	Desk calculator	
9.	11/1/24	String parser	
10.	29/1/24	Syntax tree generator	
11.	29/1/24	Infix to postfix	
12.	29/1/24	3-address code generator	



7: WALP : to count the number of vowels & consonants in a given string

```
%option noyywrap
%{
    #include<stdio.h>
    int vcount=0;
    int ccount=0;
%}
%%
a|e|i|o|u|A|E|I|O|U {vcount++;}
[a-zA-Z] {ccount++;}
\n {printf("The number of vowels in a sentence
        is:%d\n", vcount);
    printf("The number of consonents is %d\n
        ,ccount);}
%%

void main()
{ yylex()
}
```

Output:

I am Girish

The number of vowels: 4

The number of consonants: 4

Output

```
Enter a sentence:
Compiler design
 No of vowels and consonants are 5 and 9
This is a book
   No of vowels and consonants are 5 and 6
```

Pgm-9 WALP to read the following input from a file & print valid tokens on the terminal

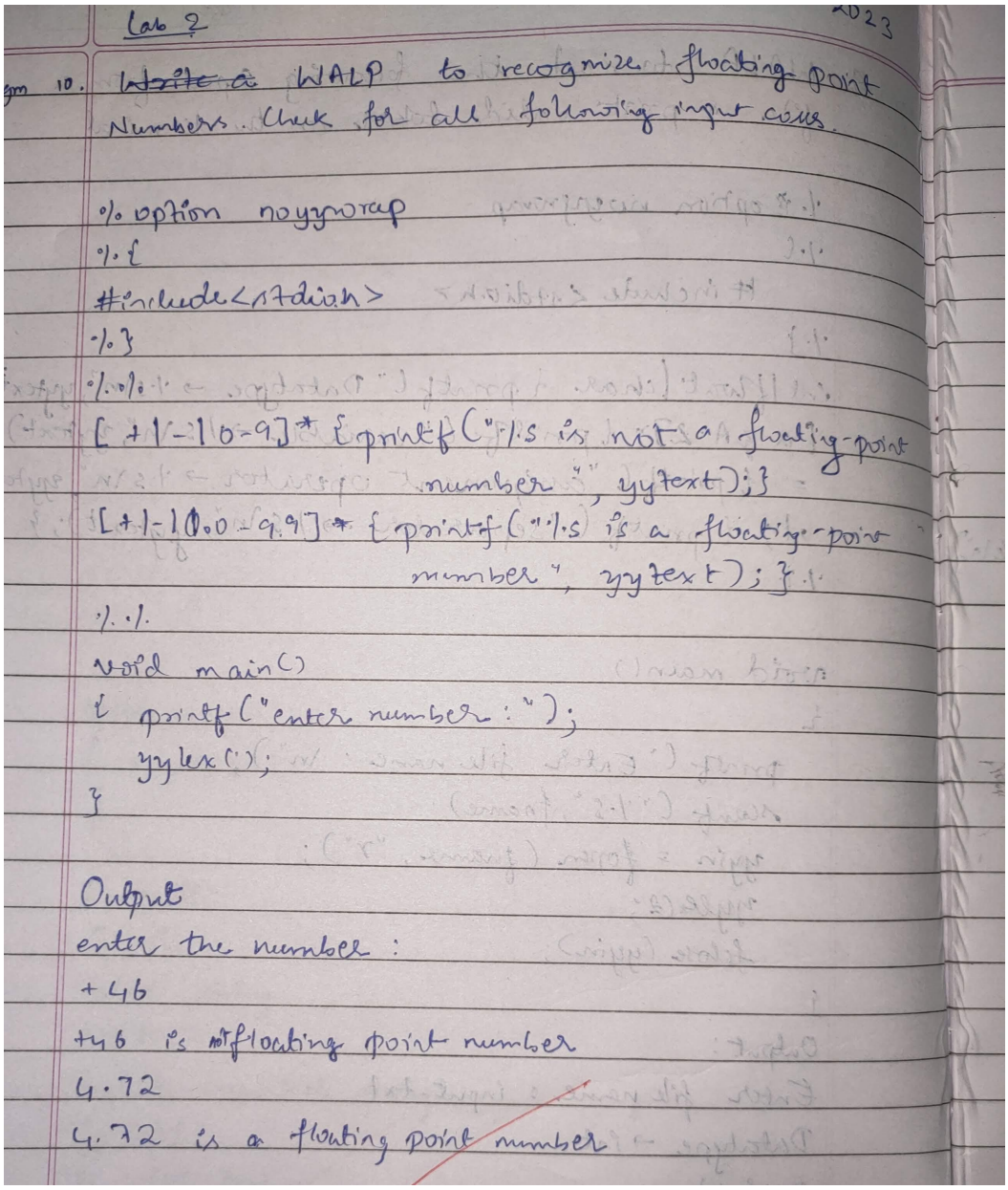
```
%{
#include<stdio.h>
%}
%option noyywrap
int|float|char {printf("Datatype → %s\n",yytext);}
[a-zA-Z]* {printf("variable → %s\n",yytext);}
= {printf("assignment operator → %s\n",yytext);}
,|; {printf("separator → %s\n",yytext);}
%%

void main()
{
  printf("Enter file name : \n");
  scanf("%s",fname);
  yyin = fopen(fname,"r");
  yylex();
  fclose(yyin);
}
```

Output:

```
Enter file name : input.txt
Datatype → int
variable → a
assignment operator → =
digit → 5
separator → ,
variable → sum
separator → ;
```

Output :

```
int is a keyword.
 sum is an identifier.
, is a separator.
x is an identifier.
= is an assignment operator.
2 is/are digit(s).
, is a separator.
y is an identifier.
= is an assignment operator.
3 is/are digit(s).
; is a delimiter.
sum is an identifier.
= is an assignment operator.
x is an identifier.
+ is a binary operator.
y is an identifier.
; is a delimiter.
```


Lab 2

10. ~~Write a~~ WALP to recognize floating point Numbers. Check for all following input cases.

```
%option noyywrap
%{
#include<stdio.h>
%}
%%
[+|-]10-9]* {printf("%s is not a floating-point number", yytext);}
[+|-]10.0-9.9]* {printf("%s is a floating-point number", yytext);}
%%
void main()
{ printf("enter number:");
  yylex();
}
```

Output

```
enter the number :
+46
+46 is not floating point number
4.72
4.72 is a floating point number
```

Output

```
Enter a number:
23
Not a floating point number!

0.5
Floating point number!

.8
Floating point number!

-.9
Floating point number!

+56
Not a floating point number!
```

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

```
%{
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
char string[200];
%}

%%
[\n] { fprintf(yyout, "%s\n", string); string[0]='\0'; }
[ ]* { fprintf(yyout, "%s", string); string[0]='\0';
       fprintf(yyout, "%s", " "); }
. strcat(string, yytext);
<<EOF>> { fprintf(yyout, "%s", string); return 0; }
%%

int main()
{
 extern FILE *yyin, *yyout;
char filename[100];
printf("This program is going copy a file,
  replacing each nonempty sequence of white
  space by a single blank.!\n Enter name
  of file to copy: \t");
scanf("%s", filename);

yyin = fopen(filename, "r");
if (yyin == NULL)
{ printf("Can't open file %s\n", filename);
  exit(0);
}
```

```
printf("Enter name of file to open for writing: ");
scanf("%s", filename);

yyout = fopen(filename, "w");
if (yyout == NULL)
{ printf("Cannot open file %s\n", filename);
  exit(1);
}

yylex();
}

int yywrap(void)
{ return 1;
}
```

Output:

This program is going to copy a file, replacing each nonempty sequence of white space by a single blank!
Enter name of file to copy: read
Enter name of file to open for writing: write

→

input

Hi, I am Girish Kumar S K
I am doing my Bachelors in Computer Science and Engineering.

output

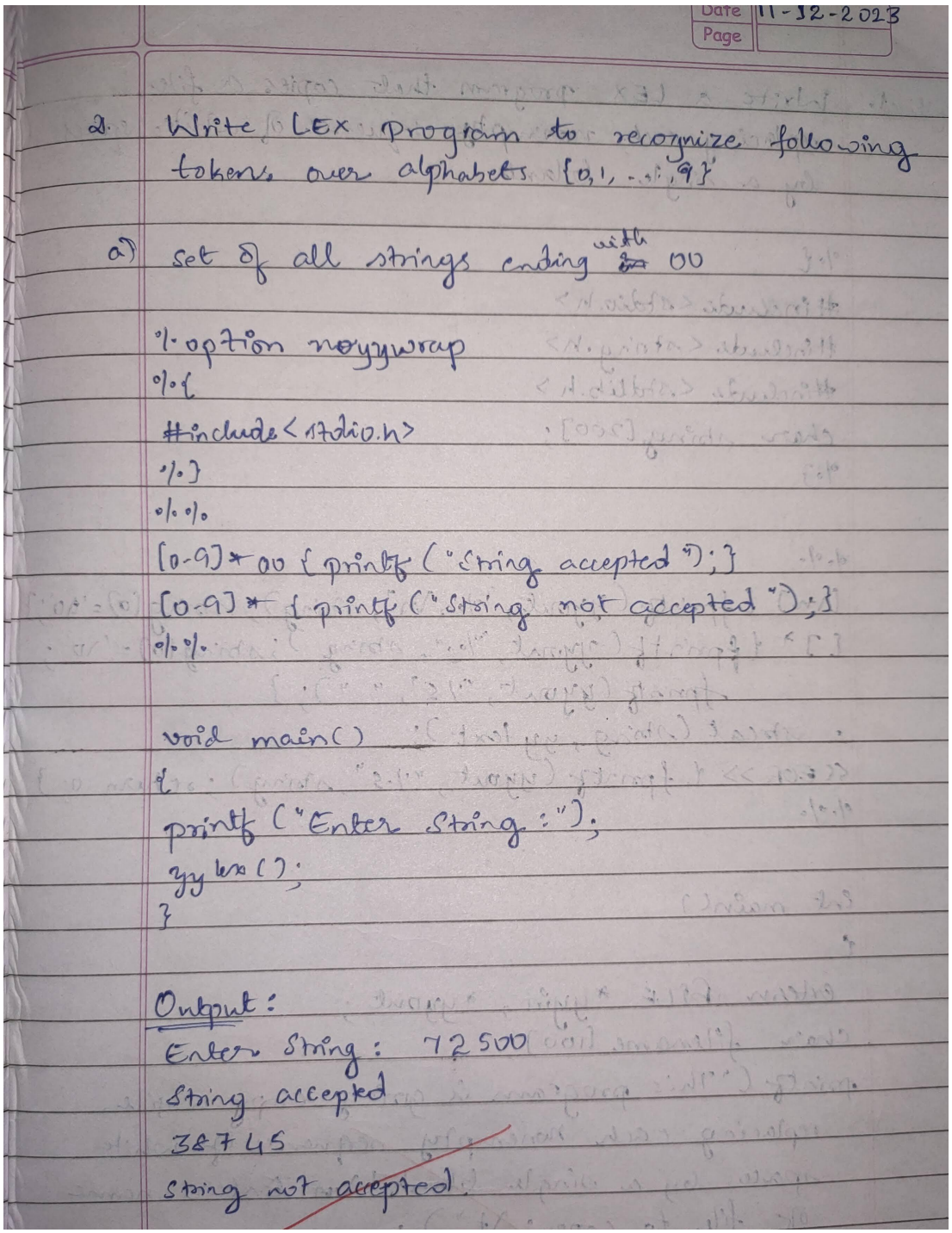
Hi, I am Girish Kumar S K
I am doing my Bachelors in Computer Scienc
and Engineering.

Output


*text.txt
1 Hello World
2 Welcome to programming


Open
print.txt
~/Documents
1 Hello World
2 Welcome to programming

4.1 Write a LEX program to recognize the following tokens over the alphabets {0,1,..,9}



Date 11-12-2023
Page

2. Write LEX program to recognize following tokens over alphabets {0,1,...,9}

a) set of all strings ending with 00

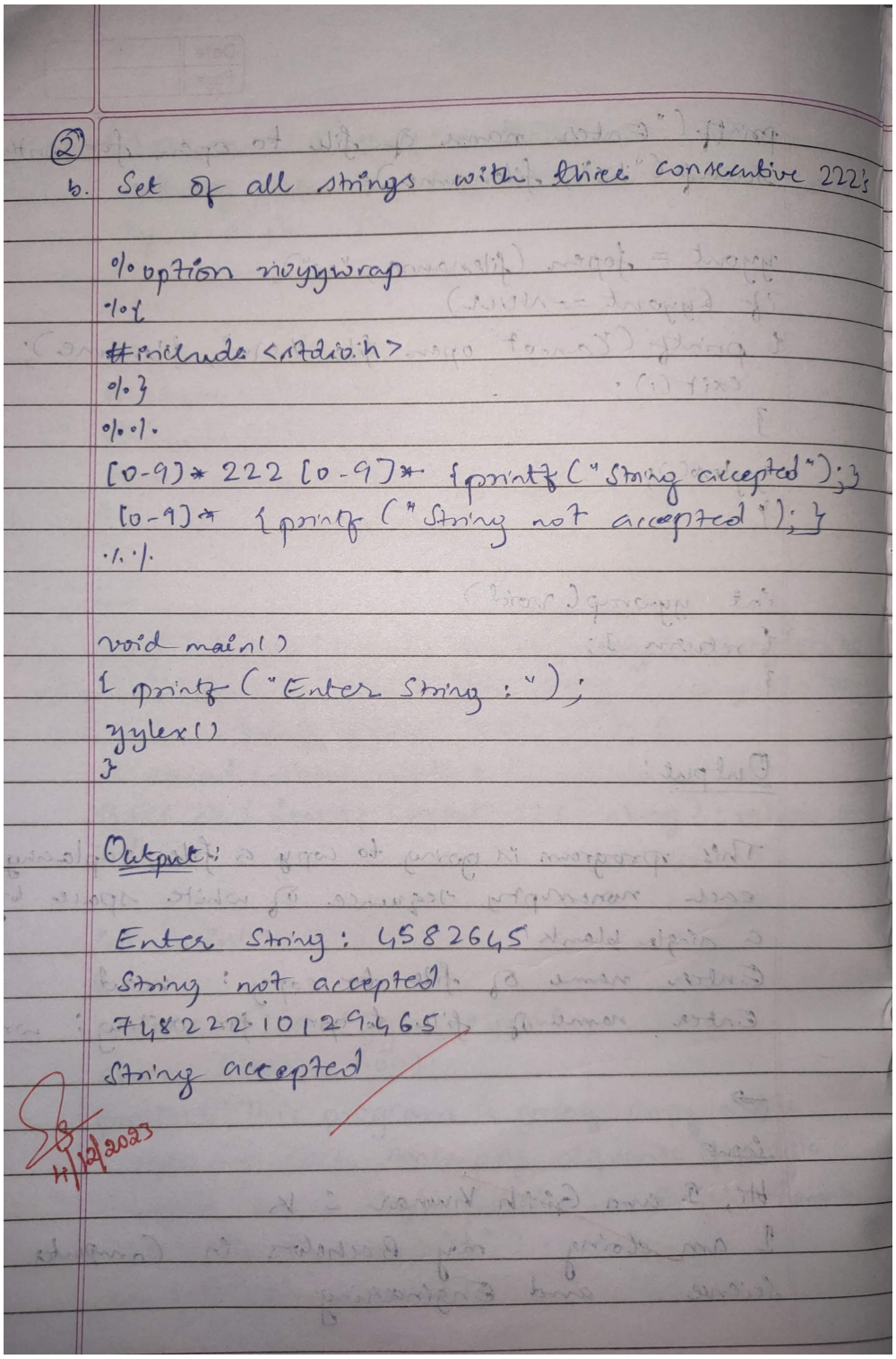
```
%.option noyywrap
%.{
#include<stdio.h>
%.}
%.%.
[0-9]*00 {printf("String accepted");}
[0-9]* {printf("String not accepted");}
%.%.

void main()
{
printf("Enter String:");
yylex();
}
```

Output:

Enter String: 72500
String accepted
3845
String not accepted.

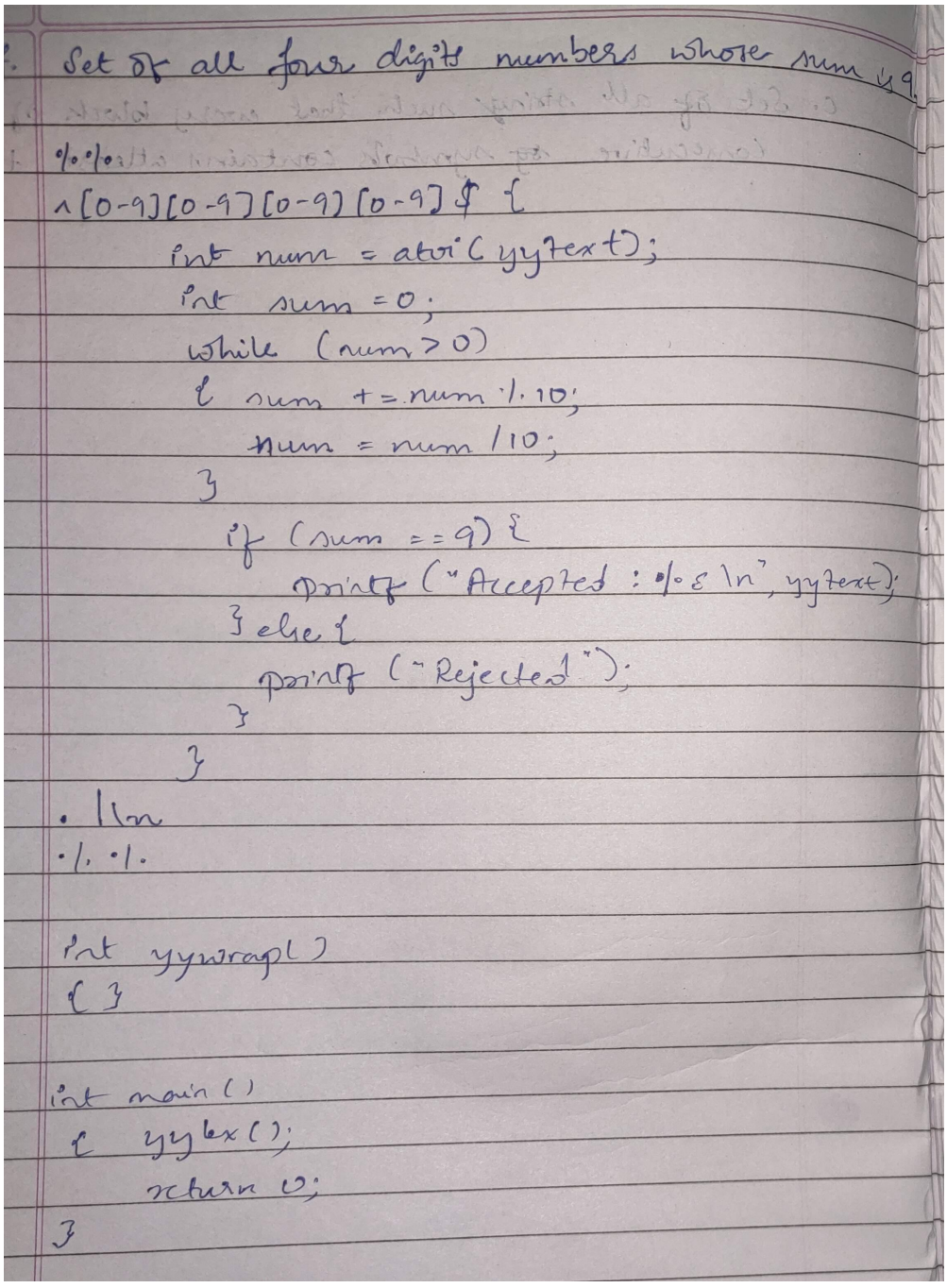
Output

```
Enter a string:
12300
Ends with 0.
```



Output

```
Enter a string:
322221
Has 3 consecutive 2's.
```



Output

```
Enter a string:
2340
The sum of digits is 9.
```

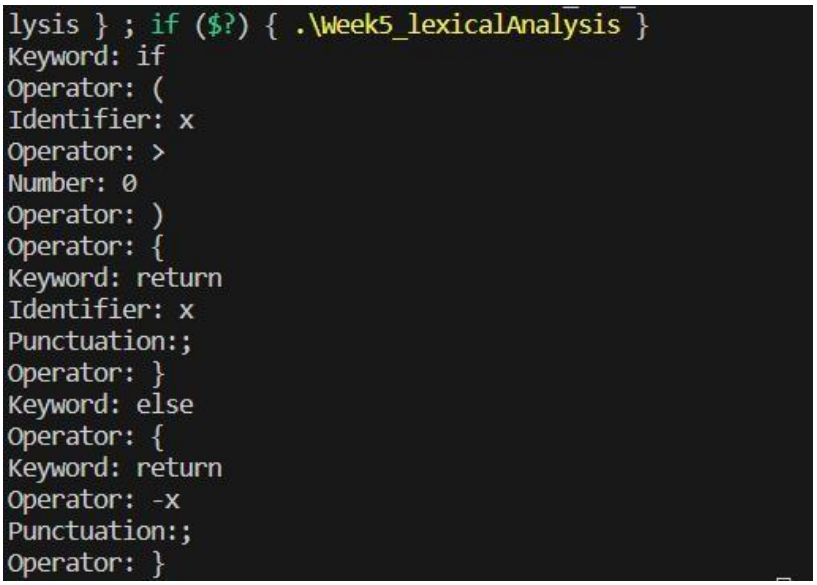
end2 =

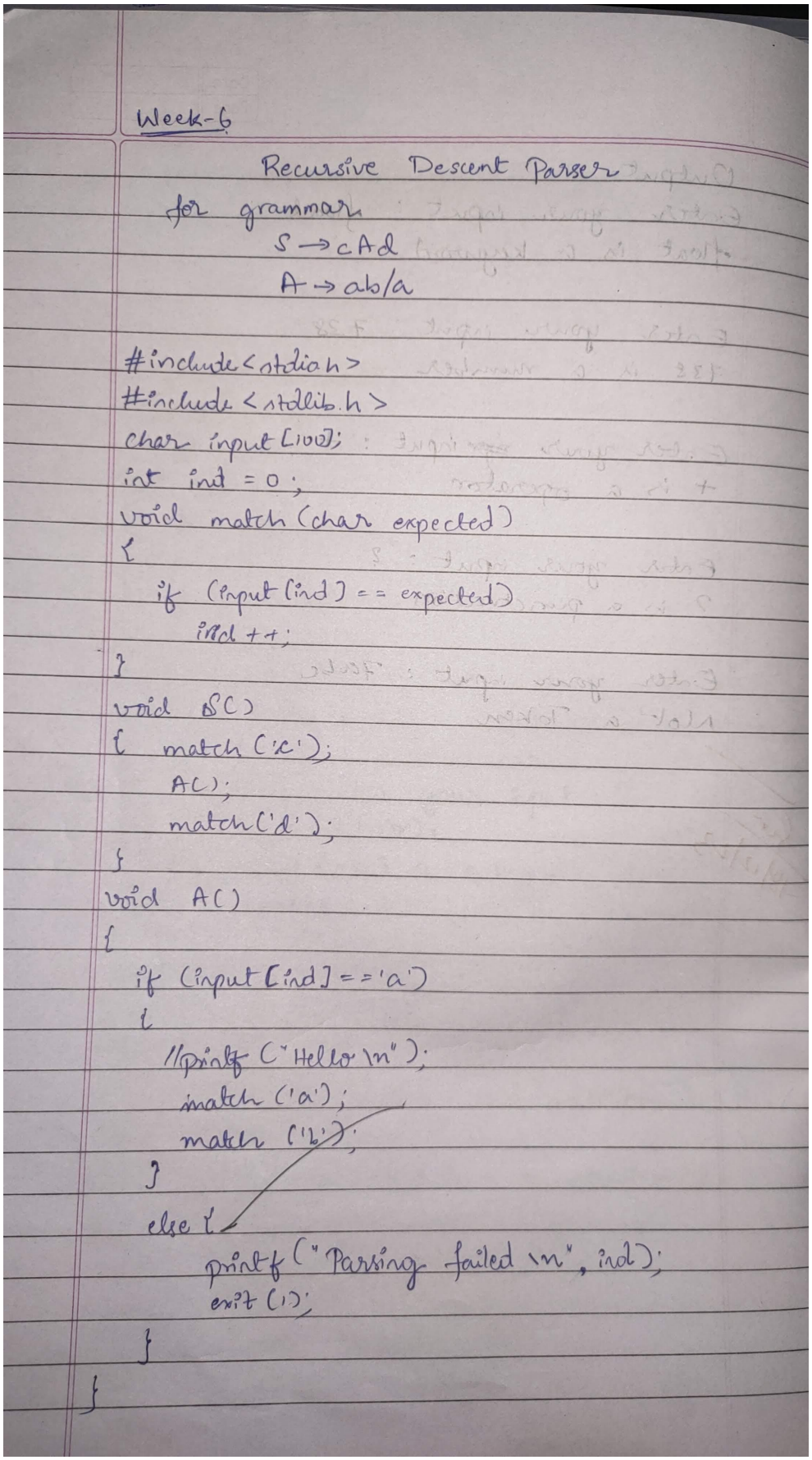
Qn. Write a python program to design Lexical Analyzer ~~in C/C++/Python~~ to recognize any five keywords, identifiers, numbers, operators & punctuation.

```
def is_keyword(str):
    if str=='int' or str=='float' or str=='if' or
       str=='else' or str=='while':
        return True
    else:
        return false

def is_number(str):
    n = len(str)
    k = 0
    for i in range(n):
        if str[i] >= '0' and str[i] <= '9':
            k = k+1
    return k==n

def is_id(str):
    if str[0]>'0' and str[0]<='9':
        return false
    n = len(str)
    k = 0
```

```
    for i in range(n):
        if (str[i]>='A' and str[i]<='Z') or
           (str[i]>='a' and str[i]<='z') or
           (str[i]>='0' and str[i]<='9') or
           (str[i]=='_'):
            k = k+1
    return k==n
    k==n
def is_op(str):
    n = len(str)
    if n>2:
        return False

    return str=='.' or str==',' or str==':'
        or str==';' or str=='!' or str=='?'

str = input("Enter your input :")
if is_keyword(str):
    print(f"{str} is a keyword")
elif is_number(str):
    print(f"{str} is a number")
elif is_id(str):
    print(f"{str} is a identifier")
elif is_op(str):
    print(f"{str} is a operator")
elif is_punc(str):
    print(f"{str} is a punctuation")
else:
    print("Enter valid input")

print("Not a token")
```

Output


lysis } ; if ($?) { .\Week5_lexicalAnalysis }
Keyword: if
Operator: (
Identifier: x
Operator: >
Number: 0
Operator:)
Operator: {
Keyword: return
Identifier: x
Punctuation:;
Operator: }
Keyword: else
Operator: {
Keyword: return
Operator: -x
Punctuation:;
Operator: }



Week-6

Recursive Descent Parser

for grammar

S → cAd

A → ab/a

```
#include<stdio.h>
#include<stdlib.h>
char input[100];
int ind = 0;
void match(char expected)
{
  if (input[ind] == expected)
    ind++;
}
void S()
{ match('c');
  A();
  match('d');
}
void A()
{
  if (input[ind] == 'a')
  {
    //printf("Hello\n");
    match('a');
    match('b');
  }
  else {
    printf("Parsing failed\n", ind);
    exit(1);
  }
}
```

```
int main()
{
    printf("Enter input string: \n");
    scanf("%s", input);

    S();
    if (input[ind] == '$')
        printf("Parsing successful.\n");
    else
        printf("Parsing failed. Extra characters found.");

    return 0;
}
```

Output:

Enter the ~~str~~ input string: cad
string is accepted by grammar

Enter the input string: cd
~~string~~ Parsing failed. ~~Extra~~

Output

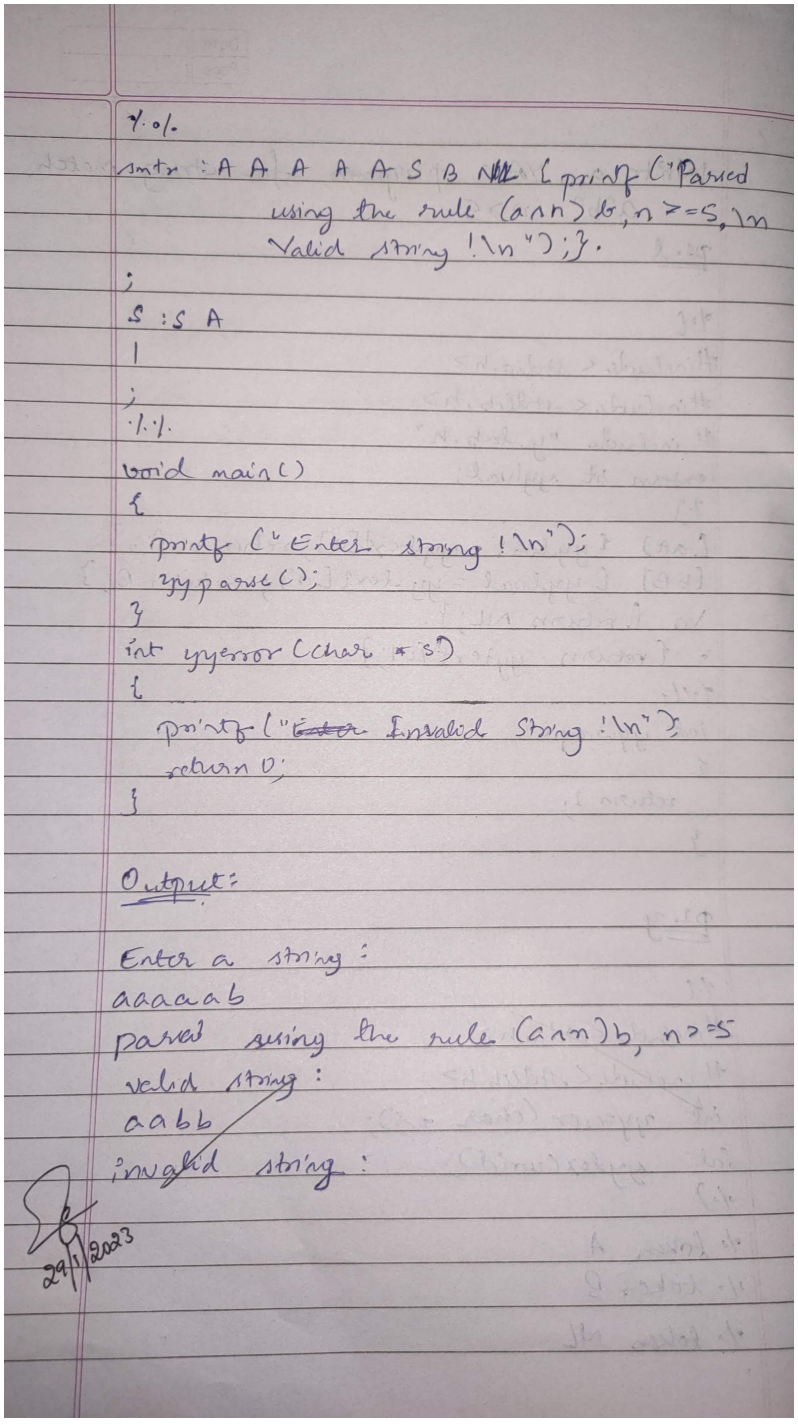
Write a YACC program for string match.
$a^n b^n$ $n \geq 5$

p1.l

```
%{
#include <stdio.h>
#include <stdlib.h>
# include "y.tab.h"
extern int yylval;
%}
[aA] { yylval = yytext[0]; return A; }
[bB] { yylval = yytext[0]; return B; }
\n { return NL; }
. { return yytext[0]; }
%%
int yywrap()
{
  return 1;
}
```

p1.y

```
%{
#include <stdio.h>
#include <stdlib.h>
int yyerror(char *s);
int yylex(void);
%}
% token A
%. token B
%. token NL
```



```
%%
stmt : A A A A A S B NL { printf("Parsed
        using the rule (a^n)b, n>=5,\n
        Valid string!\n"); }.
;
S : S A
|
;
%%
void main()
{
  printf("Enter string!\n");
  yyparse();
}
int yyerror(char *s)
{
  printf("Enter Invalid String!\n");
  return 0;
}
```

Output:

```
Enter a string:
aaaaab
parsed using the rule (a^n)b, n>=5
valid string:
aabb
invalid string:
```

29/1/2023

Output

```
Enter a string!
aaaaaaab
Parsed using the rule (a^n)b, n>=5.
Valid String!
ab
Invalid String!
```

Q. Yacc program to generate syntax tree 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 lc;
   int rc;
};
```

```
int ind;
struct tree_node syn_tree[100];
void my_print_tree(int cur_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')' { $$ = $2; }
 | digit { char buf[10]; sprintf(buf, "%d", yyval);
          $$ = mknode(-1, -1, buf); }
%%
int main()
{
  ind = 0;
  printf("Enter 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_print_tree(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);
  else
      printf("Operator Node -> Index : %d,
              Value : %s, Left Child Index : %d,
              Right Child Index : %d\n",
      cur_ind, syntree[cur_ind].val,
       syn_tree[cur_ind].lc, syntree[cur_ind].rc);
  my_print_tree(syn_tree[cur_ind].lc);
  my_print_tree(syn_tree[cur_ind].rc);
}
```

Output

```
Enter an expression:
2*3+5*4
Operator Node -> Index : 6, Value : +, Left Child Index : 2,Right Child Index : 5
Operator Node -> Index : 2, Value : *, Left Child Index : 0,Right Child Index : 1
Digit Node -> Index : 0, Value : 2
Digit Node -> Index : 1, Value : 3
Operator Node -> Index : 5, Value : *, Left Child Index : 3,Right Child Index : 4
Digit Node -> Index : 3, Value : 5
Digit Node -> Index : 4, Value : 4
```

Week - 8

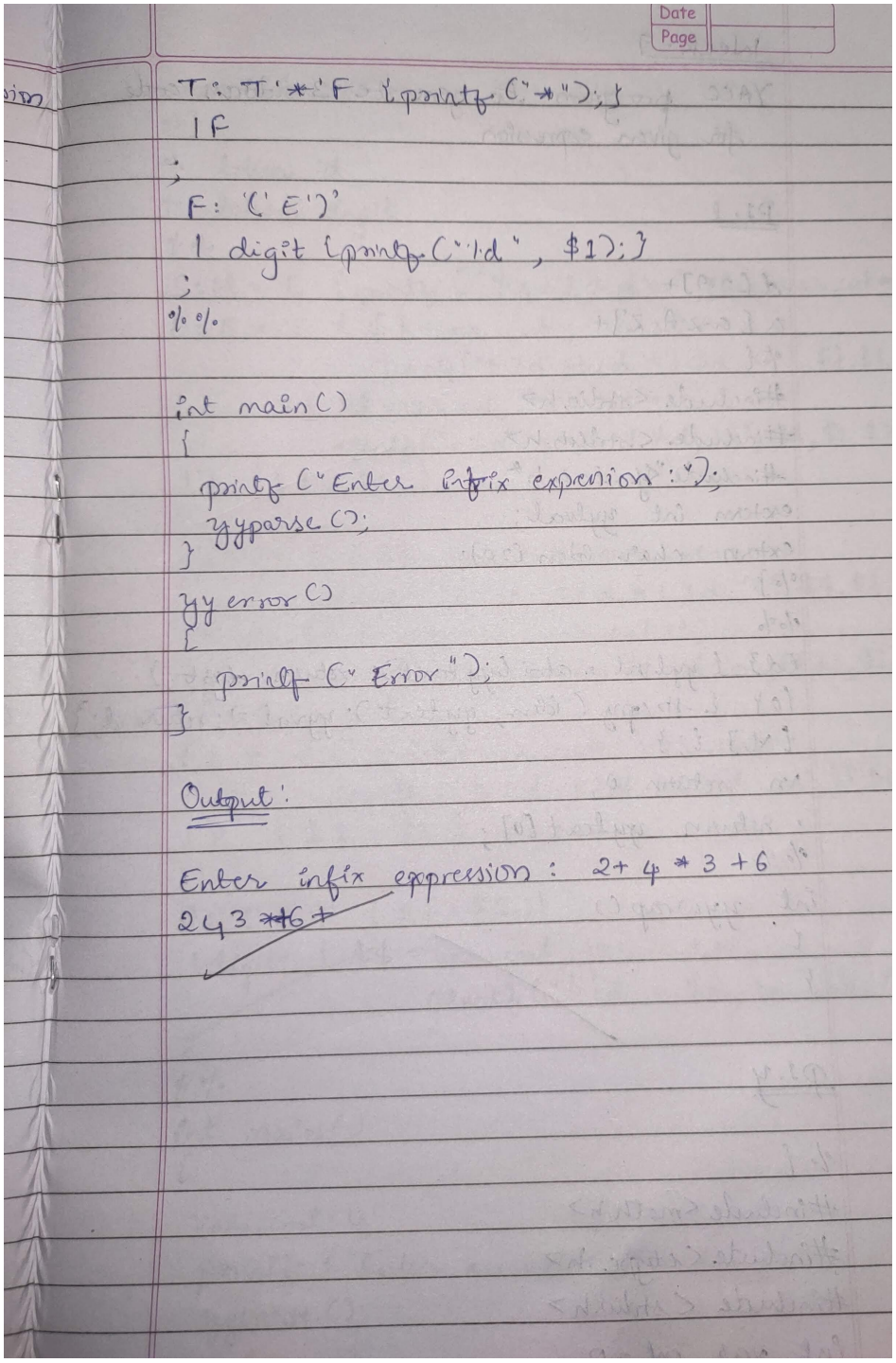
Yacc to convert infix expression to postfix expression

p4.l

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

Enter infix expression : 2+4*3+6
243*+6+

Output

```
Enter an infix expression:
2+3*8/4^3-3
238*43^/+3-
```

Week-9

YACC program to generate 3-address code for given expression

P1.l

```
d [0-9]+
a [a-zA-Z]+
%{
#include <stdio.h>
#include <stdlib.h>
#include "y.tab.h"
extern int yylval;
extern char iden[20];
%}
%%
{d} { yylval = atoi(yytext); return digit; }
{a} { strcpy(iden, yytext); yyval = 1; return id; }
[ \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 iden[20];
%}
%token id
%token digit
%%
S:id '=' E {printf("%s = t%d\n", iden, var_cnt-1);}
E: E '+' T { $$ = var_cnt; var_cnt++;
             printf("t%d = t%d + t%d;\n", $$, $1, $3);}
| E '-' T { $$ = var_cnt; var_cnt++;
             printf("t%d = t%d - t%d;\n", $$, $1, $3);}
| T { $$ = $1; }
;
T: T '*' F { $$ = var_cnt; var_cnt++;
             printf("t%d = t%d * t%d;\n", $$, $1, $3);}
| T '/' F { $$ = var_cnt; var_cnt++;
             printf("t%d = t%d / t%d;\n", $$, $1, $3);}
| F { $$ = $1; }
F: P '^' F { $$ = var_cnt; var_cnt++;
             printf("t%d = t%d ^ t%d;\n", $$, $1, $3);}
| P { $$ = $1; }
;
P: '(' E ')' { $$ = $2; }
| digit { $$ = var_cnt; var_cnt++;
             printf("t%d = %d;\n", $$, $1); }
%%
int main()
{
var_cnt = 0;
printf("Enter an expression:\n");
yyparse();
return 0;
}
```

Output


Enter an expression:
a=2*3/6-4
t0 = 2;
t1 = 3;
t2 = t0 * t1;
t3 = 6;
t4 = t2 / t3;
t5 = 4;
t6 = t4 - t5;
a=t6