

FACULTY OF ENGINEERING AND TECHNOLOGY BACHELOR OF TECHNOLOGY

Compiler Design

(CD) (203105351)

VI SEMESTER

Computer Science & Engineering Department





**CERTIFICATE**

*This is to certify that*

*Mr.*  **LUCKY PATHAN** *with Enrollment No.* **210303105790** has *successfully completed his laboratory experiments in the subject (with Code)* **Compiler Design (203105351)** *from the department of* **Computer Science and Engineering** *during the academic year* ***2022-2023.***



**Date of Submission …..…………… Staff In charge …..……………**

**Head of Department …..……………**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **S**  **r. N**  **o** | **Experiment Title** | **Page No** | | **Date of Start** | **Date of Completing** | **Sign** | **Marks**  **(Out of 10)** |
| **From** | **To** |
| 1 | Program to implement Lexical Analyzer. | 1 | 4 |  |  |  |  |
| 2 | Program to count digits, vowels and symbols in C. | 5 | 6 |  |  |  |  |
| 3 | Program to check validation of User Name and Password in C. | 7 |  |  |  |  |  |
| 4 | Program to implement Predictive Parsing LL (1) in C. |  |  |  |  |  |  |
| 5 | Program to implement Recursive Descent Parsing in C.. |  |  |  |  |  |  |
| 6 | Program to implement Operator Precedence Parsing in C. |  |  |  |  |  |  |
| 7 | Program to implement LALR Parsing in C. |  |  |  |  |  |  |
| 8 | To Study about Lexical Analyzer Generator (LEX) and Flex (Fast Lexical Analyzer) |  |  |  |  |  |  |
| 9 | Implement following programs using Lex.  a. Create a Lexer to take input from text file and count no of characters, no. of lines & no. of words.  b. Write a Lex program to count number of vowels and consonants in a given input string. |  |  |  |  |  |  |
| 10 | Implement following programs using Lex.  a. Write a Lex program to print out all numbers from the given file.  b. Write a Lex program to printout all HTML tags in file.  c. Write a Lex program which adds line numbers to the given file and display the same onto the standard output. |  |  |  |  |  |  |

# Practical 1

**Aim** : Program to implement Lexical Analyzer.

**CODE:-**

#include<stdio.h>

#include<conio.h>

#include<ctype.h>

#include<string.h>

#include<stdlib.h>

void keyw(char \*p);

int i=0,id=0,kw=0,num=0,op=0,sp =0,ar=0,count=1,new\_lune=0;

char keys[32][10]={"auto","break","case","char","const","continue","default","do","dobule","else","enum","extern","float","for","goto","if","int","long","register","return","short","signed","sizeof","struct","switch","typedef","union","unsigned","void","volatiel","while"};

void main()

{

char ch,str[25],seps[20]="\t\n,;(){}[]#\"<>",oper[]="!%^&\*-+=~|.<>/?";

int j;

char fname[50];

FILE \*f1;

f1 = fopen("Laxcode.txt","r");

if (f1 == NULL)

{

printf("File not found");

exit(0);

}

while((ch=fgetc(f1))!=EOF)

{

for ( j = 0; j <=14; j++)

{

if(ch == oper[j])

{

printf("%c is an operator\n",ch);

op++;

count++;

str[i]= '\0';

keyw(str);

}

}

if (ch == '\n')

{

new\_lune++;

}

for ( j = 0; j <= 14; j++)

{

if(i == -1)

break;

if(ch == seps[j])

{

if(ch == '#')

{

while(ch != '>')

{

printf("%c",ch);

ch = fgetc(f1);

}

printf("%c is a header file\n",ch);

i = -1;

break;

}

if (ch =='"')

{

do

{

ch = fgetc(f1);

printf("%c",ch);

} while (ch != '"');

i = -1;

ar++;

count++;

break;

}

if(ch == ',' || ch == ';' || ch == '(' || ch == ')' || ch == '{' || ch == '}' || ch == '[' || ch == ']')

{

printf("%c is an Serpator",ch);

sp++;

count++;

}

str[i]='\0';

keyw(str);

}

}

if(i!=-1)

{

str[i]=ch;

i++;

}

else

i = 0;

}

printf("\n Keywords : %d \n Identifiers : %d \n Operators : %d \n Numbers : %d \n Seprator : %d \n Arfument : %d",kw,id,op,num,sp,ar);

printf("\n Total number of token : %d",count);

printf(" \n Number of lines : %d",new\_lune);

getch();

}

void keyw(char \*p)

{

int k, flag = 0;

for (k = 0; k <=31; k++)

{

if(strcmp(keys[k],p) == 0)

{

printf("%s is a keyword \n",p);

kw++;

count++;

flag + 1;

break;

}

}

if(flag == 0)

{

if(isdigit(p[0]))

{

printf("%s us a number\n",p);

num ++;

count++;

}

else

{

if(p[0]!='\0')

{

printf("%s is a identifier\n",p);

id++;

count++;

}

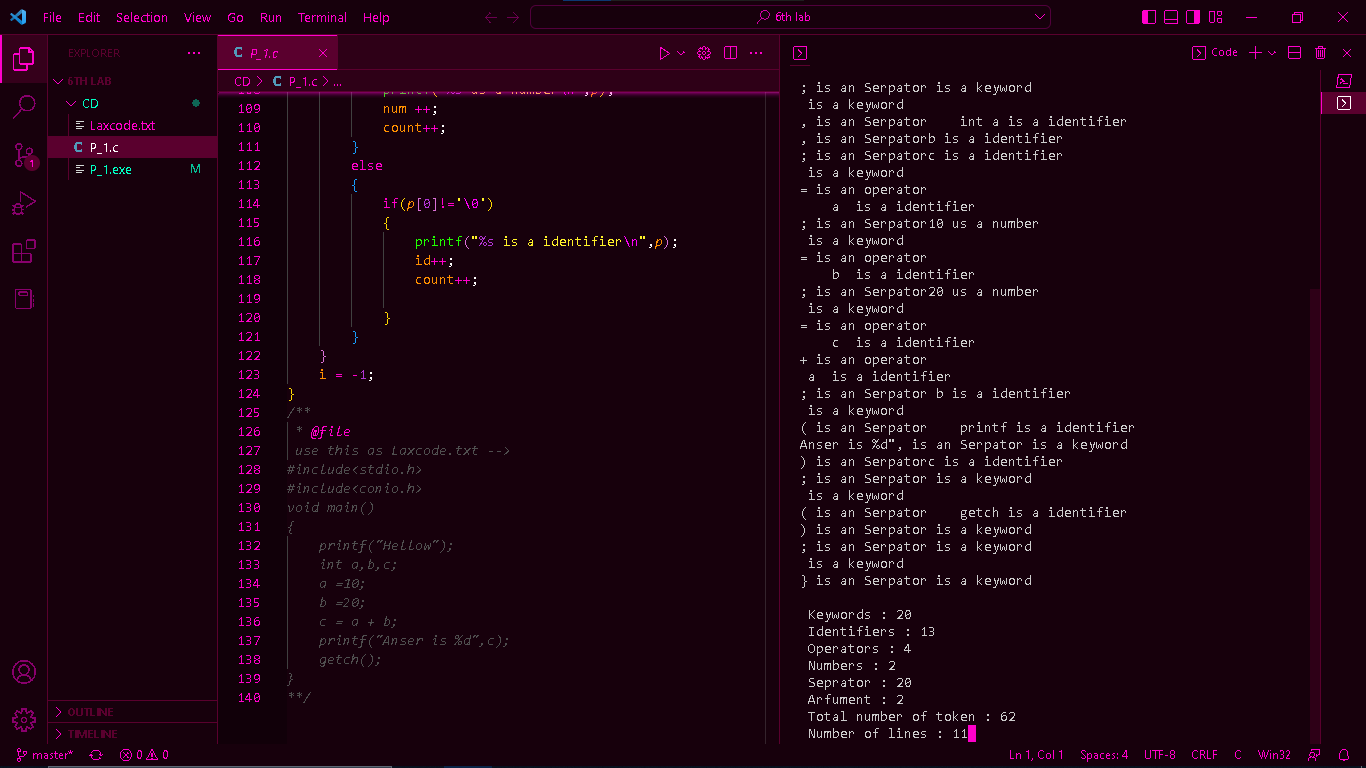
}

}

i = -1;

}

**OUTPUT:-**

****

# Practical 2

**Aim** : Program to count digits, vowels and symbols in C.

**CODE:-**

#include<stdio.h>

#include<string.h>

#include<conio.h>

#include<stdlib.h>

int main()

{

char str[100];

int i =0;

int vovels = 0 , consonant = 0 , digits = 0 , symbols = 0 , space = 0 ;

FILE \*fp;

char ch;

fp = fopen("P2.txt","r");

if(fp == NULL)

{

printf("File not opened.");

exit(1);

}

ch = fgetc(fp);

printf("your string is :\n");

while (!feof(fp))

{

str[i++]=ch;

ch = fgetc(fp);

}

str[i] = '\0';

printf(" %s",str);

fclose(fp);

for(i = 0; str[i]!=0;i++)

{

if(str[i] == 'a' || str[i] == 'A' || str[i] == 'e' ||str[i] == 'E' ||str[i] == 'i' ||str[i] == 'I' ||str[i] == 'o' ||str[i] == 'O' ||str[i] == 'u' || str[i] == 'U')

{

vovels++;

}

else if((str[i]>='a'&& str[i]<='z') || (str[i]>='Z'&& str[i]<='Z'))

{

consonant++;

}

else if ((str[i]>='0'&& str[i]<='9'))

{

digits++;

}

else if ((str[i]>=' '))

{

space++;

}

else

{

symbols++;

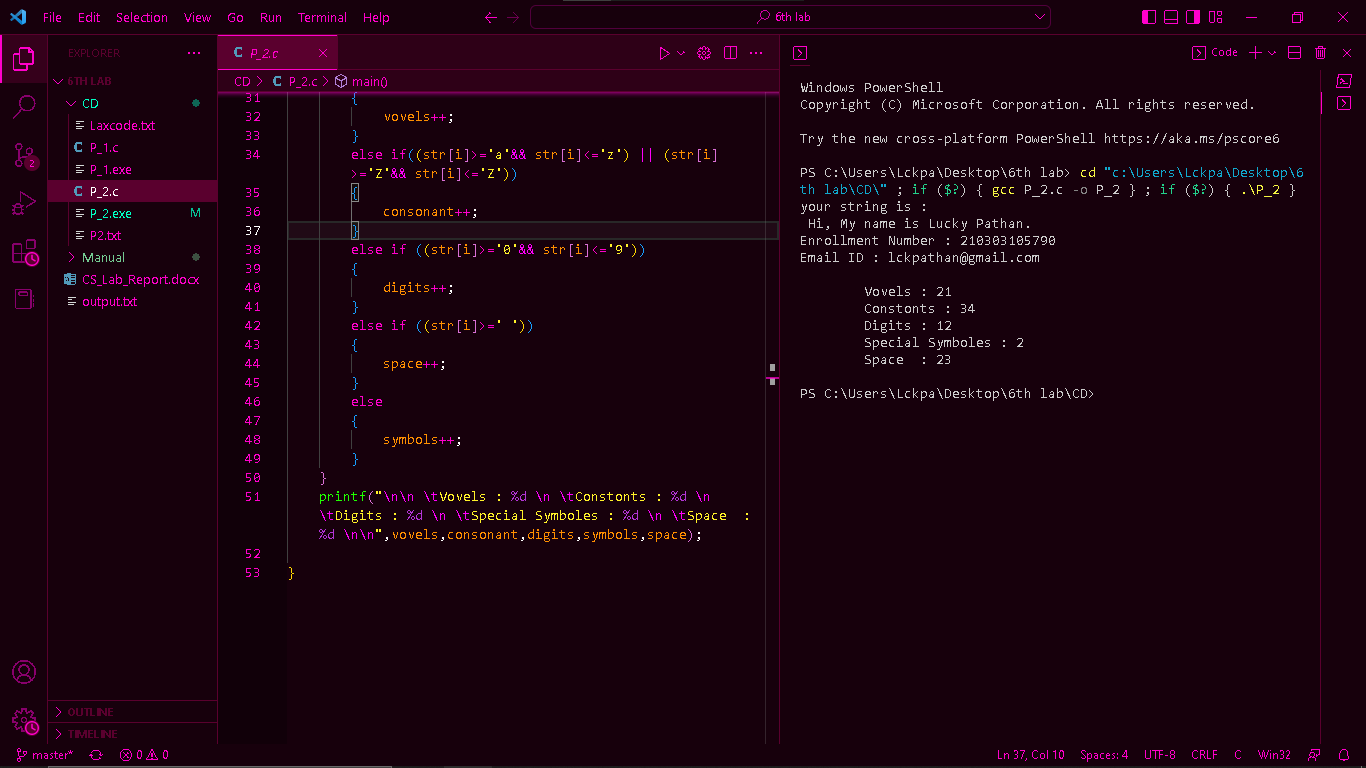
}

}

printf("\n\n \tVovels : %d \n \tConstonts : %d \n \tDigits : %d \n \tSpecial Symboles : %d \n \tSpace : %d \n\n",vovels,consonant,digits,symbols,space);

}

**OUTPUT:-**



# Practical 3

**Aim** : Program to check validation of User Name and Password in C.

**CODE:-**