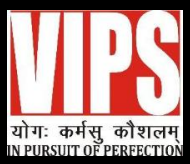
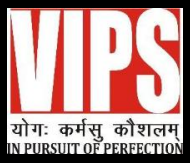
VIVEKANANDA INSTITUTE OF PROFESSIONAL STUDIES - TECHNICAL CAMPUS
A++

SCHOOL OF ENGINEERING & TECHNOLOGY



VIVEKANANDA INSTITUTE OF PROFESSIONAL STUDIES - TECHNICAL CAMPUS
A++

SCHOOL OF ENGINEERING & TECHNOLOGY



VIVEKANANDA INSTITUTE OF PROFESSIONAL STUDIES - TECHNICAL CAMPUS
A++

SCHOOL OF ENGINEERING & TECHNOLOGY

VIVEKANANDA INSTITUTE OF PROFESSIONAL STUDIES - TECHNICAL CAMPUS

Grade A++ Accredited Institution by NAAC

NBA Accredited for MCA Programme; Recognized under Section 2(f) by UGC;
Affiliated to GGSIP University, Delhi; Recognized by Bar Council of India and AICTE
An ISO 9001:2015 Certified Institution

SCHOOL OF ENGINEERING & TECHNOLOGY

INDEX

S.No	EXP.	Date	Marks			Remark	Updated Marks	Faculty Signature
			Laboratory Assessment (15 Marks)	Class Participation (5 Marks)	Viva (5 Marks)			
1.	Write a Lexical Analyzer program that identifies any 10 keywords from C language and identifiers following all the naming conventions of the C program.							
2.								
3.	Write a Syntax Analyzer program using Yacc tool that will have grammar rules for the operators : *,/,%.							
4.								

5. To write a C program that takes the single line production rule in a string as input and checks if it has Left-Factoring or not and give the unambiguous grammar, in case, if it has Left-Factoring.

```
PS C:\Users\Geetansh\Desktop\00517702722_Geetansh> cd "
hello identifiers
hello
hello
hello hello
identifiers
5
constants
5abc
constantsidentifiers
float r
keywords identifiers
```

```
PS C:\Users\Geetansh\Desktop\00517702722_Geetansh> .\1_output.exe
boolean
keywords
float
keywords
int
keywords
if
keywords
char
keywords
12
constants
Geetansh
identifiers
Geetansh CSE A
identifiersidentifiersidentifiers
THISisTeXt
identifiers
```

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    // Path to input file
    const string inputFilePath = "input.txt";
```

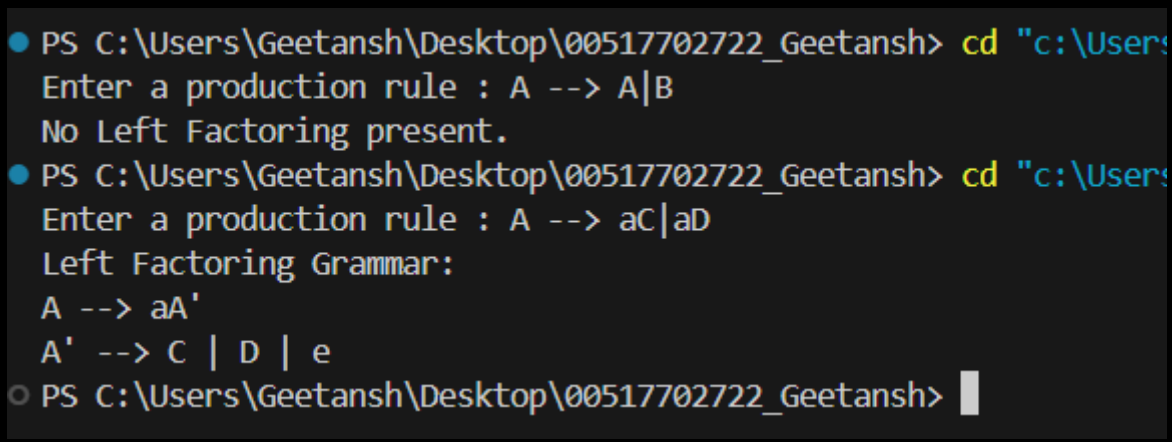
```
    // Load keywords
    unordered_set<string> keywords = {"auto", "break", "case", "char", "const", "continue",
"default", "do",
        "double", "else", "enum", "extern", "float", "for", "goto", "if",
        "int", "long", "register", "return", "short", "signed", "sizeof",
        "static", "struct", "switch", "typedef", "union", "unsigned", "void",
        "volatile", "while"};
    // Create a map to store keyword counts
    unordered_map<string, int> keywordCount;


    // Read the input file
    ifstream inputFile(inputFilePath),ss(inputFilePath);
    if (!inputFile) {
        cerr << "Error opening file: " << inputFilePath << endl;
        return 1;
    }
       string token;
        while (inputFile >> token) {
            if (keywords.find(token) != keywords.end()) {
                keywordCount[token]++;
            }
        }
        // output the string in file
        cout << "Content of " << inputFilePath<<":   ";
        while (ss >> token) {
            cout<<token;
        }
        cout<<endl;
        for (const auto& pair : keywordCount) {
            if (pair.second > 0) {
                cout << pair.first << ": " << pair.second << endl;
            }
```

```
    }
    inputFile.close();
  return 0;
}
```

```
PS C:\Users\Geetansh\Desktop\00517702722_Geetansh> c
Content of input.txt:   #include<stdio.h>intmain(){i
return: 1
int: 2
PS C:\Users\Geetansh\Desktop\00517702722_Geetansh>
```

```
PS C:\Users\Geetansh\Desktop\00517702722_Geetansh> cd "c:\Users\Geetansh
a=5
Stored: A = 5
b=10
1 0
Stored: B = 10
a+b
15
a-b
-5
a*b
50
a/b
0
a%b
5
a/0
error: division by zero
5
a==b
error: syntax error
error: invalid statement
PS C:\Users\Geetansh\Desktop\00517702722_Geetansh>
```

```
PS C:\Users\Geetansh\Desktop\00517702722_Geetansh> cd "c:\Users\Geetansh\Desktop\00517
Enter the production rule : B->a|b
No Left Recursion detected.
PS C:\Users\Geetansh\Desktop\00517702722_Geetansh> cd "c:\Users\Geetansh\Desktop\00517
Enter the production rule : A->Aa|b
Left Recursive Grammar Detected.
A -> bA'
A' -> aA' | e
PS C:\Users\Geetansh\Desktop\00517702722_Geetansh>
```


PS C:\Users\Geetansh\Desktop\00517702722_Geetansh> cd "c:\Users
Enter a production rule : A --> A|B
No Left Factoring present.
PS C:\Users\Geetansh\Desktop\00517702722_Geetansh> cd "c:\Users
Enter a production rule : A --> aC|aD
Left Factoring Grammar:
A --> aA'
A' --> C | D | e
PS C:\Users\Geetansh\Desktop\00517702722_Geetansh>

```
PS C:\Users\Geetansh\Desktop\00517702722_Geetansh> cd "c:\Users\G
22_Geetansh\" ; if ($?) { g++ a.cpp -o a } ; if ($?) { .\a }
Enter the number of productions: 2
Enter the productions (e.g., S-->aA|b):
S-->aA|e
B-->p|q
FIRST sets:
FIRST(B) = { p q }
FIRST(S) = { a epsilon }
PS C:\Users\Geetansh\Desktop\00517702722_Geetansh>
```

```
PS C:\Users\Geetansh\Desktop\00517702722_Geetansh> cd "c:\Users\Geetansh\Des
22_Geetansh\" ; if ($?) { g++ a.cpp -o a } ; if ($?) { .\a }
Enter the number of productions: 2
Enter the productions (e.g., S-->aA|b):
S-->aA|b
A-->c|d
Enter the input string: ac
Shift: c | Stack: a
Shift:  | Stack: c
Reduce: c -> A
Reduce: aA -> S
Accepted
PS C:\Users\Geetansh\Desktop\00517702722_Geetansh> cd "c:\Users\Geetansh\Des
22_Geetansh\" ; if ($?) { g++ a.cpp -o a } ; if ($?) { .\a }
Enter the number of productions: 2
Enter the productions (e.g., S-->aA|b):
S-->aA|b
A-->c|d
Enter the input string: av
Shift: v | Stack: a
Shift:  | Stack: v
Not Accepted
PS C:\Users\Geetansh\Desktop\00517702722_Geetansh>
```

```
PS C:\Users\Geetansh\Desktop\00517702722_Geetansh> cd "c:\Users\Geetansh\Desktop
22_Geetansh\" ; if ($?) { g++ a.cpp -o a } ; if ($?) { .\a }
Enter grammar productions (type 'end' to finish):
S-->a+b
A-->c
B-->d
end
The grammar is operator precedence.
PS C:\Users\Geetansh\Desktop\00517702722_Geetansh> cd "c:\Users\Geetansh\Desktop
22_Geetansh\" ; if ($?) { g++ a.cpp -o a } ; if ($?) { .\a }
Enter grammar productions (type 'end' to finish):
S-->a+b
A-->c-d
B-->e
end
The grammar is not operator precedence.
PS C:\Users\Geetansh\Desktop\00517702722_Geetansh>
```