

Compiler Design

3170701

Semester:-7
Lab Manual

Submitted By:

DISHEN MAKWANA
180470107035



Department of Computer Engineering
V.V.P. Engineering College

Vision of Institute

To be an exemplary institute, transforming students into competent professionals with human values.

Mission of Institute

- To provide a conducive academic environment for strengthening technical capabilities of the students.
- To strengthen linkage with industries, alumni and professional bodies.
- To organise various co-curricular and extra-curricular activities for overall development of the students.
- To practice good governance and conduct value- based activities for making students responsible citizens.

Vision of Department

Transforming students into globally efficient professionals with moral values.

Mission of Department

- To provide a strong foundation of computer engineering through effective teaching learning process.
- To enhance industry linkage & alumni network for better placement and real-world exposure.
- To provide various opportunities and platforms for all round development of students & encourage them for value-based practices.

Program Educational Objectives PEOs

Graduates will be able to :

- Apply computer engineering theories, principles and skills to meet the challenges of the society.
- Communicate effectively, work collaboratively and manifest professionalism with ethics.
- Exhibit life-long learning attitude and adapt to rapid technological changes in industry.
- Advance their career in industry, pursue higher education or become an entrepreneur.

Course Outcomes

Students will be able to

- Describe the basic concepts of automata theory, formal languages, and Compilation Process. (U)
- Apply suitable parsing strategies for compiling a program. (AP)
- Describe intermediate code generation, Code optimization and Error Recovery mechanisms. (U)
- Understand issues of run time environments and scheduling for instruction level parallelism. (U)



V.V.P. ENGINEERING COLLEGE

RAJKOT

Certificate

This is certify that

Mr. DISHEN MAKWANA, Enrollment No:- 180470107035 Branch:-
COMPUTER ENGINEERING, Semester:- 7 has satisfactory completed the
course in the subject:- Compiler Design (3170701) within the four walls
of V.V.P. ENGINEERING COLLEGE, Rajkot.

Date of Submission :-

Dr. Kamal Sutaria,
Staff In-Charge

Head of Department,
Department of Computer Engineering,
V.V.P. Engineering College

Index

1	Practical:-1 Implement following DOS commands. 1.COPY	6
2	Practical:-2 Implement following DOS commands. 1.MKDIR and 2.RMDIR	8
3	Practical:-3 Implement following DOS commands. 3. DIR	10
4	Practical:-4 Implement a program to implement simple lexical analyzer.	14
5	Program:-5 Write a program to eliminate left recursion from a grammar.	19
6	Program:-6 Write a program to left factor a grammar.	22
7	Program:-7 Write a program that will find the FIRST SET of the grammar.	25
8	Program:-8 Write a program that will find the FOLLOW SET of the grammar.	29
9	Program:-9 Implement RDP for a grammar.	34
10	Program:- 10 To Study about Lexical Analyzer Generator(LEX) and Flex(Fast Lexical Analyzer)	39
11	Program:-11 A lexer to print out all numbers from a given file. (Hint: By default lex reads from standard input).	41
12	Program:-12 A lexer which adds line numbers to the given file and display the same onto the standard output.	43
13	Program:-13 A lexer, which classifies tokens as words, numbers or "other".	44

1 Practical:-1 Implement following DOS commands. 1.COPY

Program:

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  int32_t main()
5  {
6      string source, dest, line;
7      cin >> source >> dest;
8
9      ifstream in(source, ios::in);
10     ofstream out(dest, ios::out);
11
12     if (!in.is_open())
13     {
14         cout << "file not found";
15     }
16     else
17     {
18         out << in.rdbuf();
19     }
20
21     string words;
22     fstream file;
23
24     file.open(dest);
25     while (file >> words)
26     {
27         cout << words << " ";
28     }
29
30     in.close();
31     out.close();
32     return 0;
33 }
```

OUTPUT:

The screenshot shows the Visual Studio Code interface. On the left, the Explorer pane shows a project structure with files like `cp_song.txt`, `cp.cpp`, `cp.exe`, `song.txt`, and a folder named `mkdir`. The main editor displays the contents of `cp.cpp`, which is a C++ program that reads from a source file and writes to a destination file. The program includes `<bits/stdc++.h>` and uses the `std` namespace. The `main` function takes an `int32_t` argument and uses `ifstream` and `ofstream` to handle file operations. It checks if the source file is open and prints "file not found" if it is not. It then reads the source file line by line and writes it to the destination file. Finally, it opens the destination file and writes the contents of the source file to it.

```

1 #include "bits/stdc++.h"
2 using namespace std;
3
4 int32_t main()
5 {
6     string source, dest, line;
7     cin >> source >> dest;
8
9     ifstream in(source, ios::in);
10    ofstream out(dest, ios::out);
11
12    if (in.is_open())
13    {
14        cout << "file not found";
15    }
16    else
17    {
18        out << in.rdbuf();
19    }
20
21    string words;
22    fstream file;
23
24    file.open(dest);
25    while (file >> words)

```

The bottom pane shows the terminal output. It displays the copyright notice for Microsoft Corporation, a PowerShell update notification, and the command prompt showing the execution of the program. The command prompt shows the command `cd "d:\CD\cp\" ; if ($?) { g++ cp.cpp -o cp } ; if ($?) { .\cp } song.txt cp_song.txt` and the output `PS D:\CD\cp> |`.

Answer the following Questions based on Program.

1. How to pass run time arguments in program?

Answer: The run time arguments can be passed while executing the program. The arguments need to be passed along with the name of the program.

2. Which error detects by your program if source file doesn't exist?

Answer: If the source file does not exist, the program will raise "file not exist!" error.

3. What if destination file is not available, is program execute successfully?

Answer: If the destination file is not available, a new file will be created with the name provided and the program will execute successfully.

2 Practical:-2 Implement following DOS commands. 1.MKDIR and 2.RMDIR

Program:

```
1  #include <bits/stdc++.h>
2  #include <iostream>
3  using namespace std;
4
5  int32_t main(int argc, char *argv[])
6  {
7      if (argv[1] == "mkdir")
8      {
9          int x = mkdir(argv[2]);
10         cout << x << endl;
11         if (x == -1)
12         {
13             cout << "Directory created" << endl;
14             return 0;
15         }
16         else
17         {
18             cout << "Error" << endl;
19             return 0;
20         }
21     }
22     else if (argv[1] == "rmdir")
23     {
24         int x = rmdir(argv[2]);
25         cout << x << endl;
26         if (x == -1)
27         {
28             cout << "Directory deleted" << endl;
29             return 0;
30         }
31         else
32         {
33             cout << "Error" << endl;
34             return 0;
35         }
36     }
37     return 0;
38 }
```

OUTPUT:

The screenshot shows a Visual Studio Code editor with a C++ file named `mkdir.cpp`. The code is as follows:

```

1  #include <bits/stdc++.h>
2  #include <iostream>
3  using namespace std;
4
5  int32_t main(int argc, char *argv[])
6  {
7      if (argv[1] == "mkdir")
8      {
9          int x = mkdir(argv[2]);
10         cout << x << endl;
11         if (x == -1)
12         {
13             cout << "Directory created" << endl;
14             return 0;
15         }
16         else
17         {
18             cout << "Error" << endl;
19             return 0;
20         }
21     }
22     else if (argv[1] == "rmdir")
23     {
24         int x = rmdir(argv[2]);
25         cout << x << endl;

```

The terminal output shows the command prompt running the program with the following commands and results:

```

PS D:\CD> cd "d:\CD\mkdir\" ; if ($?) { g++ mkdir.cpp -o mkdir } ; if ($?) { .\mkdir } rmdir cdkks
PS D:\CD\mkdir> cd "d:\CD\mkdir\" ; if ($?) { g++ mkdir.cpp -o mkdir } ; if ($?) { .\mkdir } mkdir cdkks

Directory: D:\CD\mkdir

Mode                LastWriteTime         Length Name
----                -
d-----            30-06-2021 10:55 AM             cdkks

```

Answer the following Questions based on Program.

1. What is the return type of MKDIR() and RMDIR()?

Answer: The return type of MKDIR() and RMDIR() is integer. It returns Output successful completion and -1 in case of any error.

2. Which error detects by your program if source Directory exist?

Answer: It display "error : error message" like this.

3. What will be the error for RMDIR(), if directory doesn't available ?

Answer: If directory isn't available, the error for RMDIR() will be "Error : error message".

3 Practical:-3 Implement following DOS commands. 3. DIR

Program:

```

1  #include <unistd.h>
2  #include <sys/stat.h>
3  #include <sys/types.h>
4  #include <time.h>
5  #include <string.h>
6  #include <dirent.h>
7  #include <iostream>
8  #include <iomanip>
9  #include <pwd.h>
10 using namespace std;
11
12 //Print file permissions
13 void printPriority(struct stat statinfo)
14 {
15     mode_t mode = statinfo.st_mode;
16     //Judging the file type
17     cout << (S_ISDIR(mode) ? 'd' : '-');
18
19     //DetermineUSR permissions
20     cout << (mode & S_IRUSR ? 'r' : '-');
21     cout << (mode & S_IWUSR ? 'w' : '-');
22     cout << (mode & S_IXUSR ? 'x' : '-');
23
24     //DetermineGRP permissions
25     cout << (mode & S_IRGRP ? 'r' : '-');
26     cout << (mode & S_IWGRP ? 'w' : '-');
27     cout << (mode & S_IXGRP ? 'x' : '-');
28
29     //Judgment of OTH authority
30     cout << (mode & S_IROTH ? 'r' : '-');
31     cout << (mode & S_IWOTH ? 'w' : '-');
32     cout << (mode & S_IXOTH ? 'x' : '-');
33
34     cout << " ";
35 }
36
37 //Print owner and group
38 void printOwner(struct stat statinfo)
39 {
40     struct passwd *passwdinfo = getpwuid(statinfo.st_uid);
41     cout << passwdinfo->pw_name << " ";
42     passwdinfo = getpwuid(statinfo.st_gid);
43     cout << passwdinfo->pw_name << " ";
44 }
45
46 //Print file size
47 void printSize(struct stat statinfo)
48 {
49     cout << setw(6) << statinfo.st_size << " ";
50 }
51
52 //Print Time
53 void printTime(struct stat statinfo)
54 {

```

```

55     time_t rawtime = statinfo.st_mtime;
56     struct tm *timeinfo = localtime(&rawtime);
57     cout << timeinfo->tm_mon + 1 << "Month" << timeinfo->tm_mday << "Day";
58     if (timeinfo->tm_hour < 9)
59         cout << "0" << timeinfo->tm_hour << ":";
60     else
61         cout << timeinfo->tm_hour << ":";
62
63     if (timeinfo->tm_min < 9)
64         cout << "0" << timeinfo->tm_min << " ";
65     else
66         cout << timeinfo->tm_min << " ";
67 }
68
69 void printName(const char *name)
70 {
71     cout << name << " ";
72 }
73
74 void ls_l(const char *path)
75 {
76     chdir(path);
77     DIR *dir = opendir(".");
78     struct dirent *dirinfo;
79     while (dirinfo = readdir(dir))
80     {
81         if (!strcmp(dirinfo->d_name, ".") || !strcmp(dirinfo->d_name, ".."))
82             continue;
83         struct stat statinfo;
84         stat(dirinfo->d_name, &statinfo);
85         //begin
86         printPriority(statinfo);
87         printOwner(statinfo);
88         printSize(statinfo);
89         printTime(statinfo);
90         printName(dirinfo->d_name);
91         //end
92         cout << endl;
93     }
94 }
95
96 int main()
97 {
98     ls_l("/home/wwx/Linux/Day04/fileSystem");
99     return 0;
100 }

```

OUTPUT:

```

36
37 //Print owner and group
38 void printOwner(struct stat statinfo)
39 {
40     struct passwd *passwdinfo = getpwuid(statinfo.st_uid);
41     cout << passwdinfo->pw_name << " ";
42     passwdinfo = getpwuid(statinfo.st_gid);
43     cout << passwdinfo->pw_name << " ";
44 }
45
46 //Print file size
47 void printSize(struct stat statinfo)
48 {
49     cout << setw(6) << statinfo.st_size << " ";
50 }
51
52 //Print Time
53 void printTime(struct stat statinfo)
54 {
55     time_t rawtime = statinfo.st_mtime;
56     struct tm *timeinfo = localtime(&rawtime);
57     cout << timeinfo->tm_mon + 1 << "Month" << timeinfo->tm_mday << "Day";
58     if (timeinfo->tm_hour < 9)
59         cout << "0" << timeinfo->tm_hour << ":";
60     else
61         cout << timeinfo->tm_hour << ":";
62
63     if (timeinfo->tm_min < 9)
64         cout << "0" << timeinfo->tm_min << " ";
65     else
66         cout << timeinfo->tm_min << " ";
67 }
68
69 void printName(const char *name)
70 {
71     cout << name << " ";
72 }

```

```

1 www@VM-0-7-ubuntu:~/Linux/Day04/homework5$ cd "/home/www/Linux/Day04/homework5/" && g++ main.cpp -o m
2 -rw-rw-r-- www www 3178 May 18th 13:45 main1.cpp
3 -rwxr-xr-x www www 1035 May 14 20:01 file3
4 -rw-rw-r-- www www 2917 May 18th 13:53 main2.cpp
5 -rw-rw-r-- www www 1000 May 14 14:47 file1
6 -rw-rw-r-- www www 11 May 14 20:01 file4
7 -rwxr-xr-x www www 17 May 14 20:19 file6
8 -rwxr-xr-x www www 4192 May 14 14:43 file2
9 -rwxrwxr-x www www 14272 May 18th 13:45 main1
10 drwxrwxr-x www www 4096 May 14 10:06 newdir

```

```

1 www@VM-0-7-ubuntu:~/Linux/Day04/homework5$ ls -l /home/www/Linux/Day04/fileSystem
2 total 52
3 -rw-rw-r-- 1 www www 1000 May 14 14:47 file1
4 -rwxr-xr-x 1 www www 4192 May 14 14:43 file2
5 -rwxr-xr-x 1 www www 1035 May 14 20:01 file3
6 -rw-rw-r-- 1 www www 11 May 14 20:01 file4
7 -rwxr-xr-x 1 www www 17 May 14 20:19 file6
8 -rwxrwxr-x 1 www www 14272 May 18 13:45 main1
9 -rw-rw-r-- 1 www www 3178 May 18 13:45 main1.cpp
10 -rw-rw-r-- 1 www www 2917 May 18 13:53 main2.cpp
11 drwxrwxr-x 4 www www 4096 May 14 10:06 newdir

```

Answer the following Questions based on Program.

1. Why the number of files are displayed more as compare to original?

Answer: Even if we think that those two commands would give us the same output, it is not actually true. When running "DIR" with the "-l" option, you are also printing a line for the total disk allocation for all files in this directory. As a consequence, you are counting a line that should not be counted, incrementing the final result by one.

2. What is the meaning of . and .. in output list?

Answer: . means current directory,
.. means parent directory.

3. What if destination file is not available, will program execute successfully?

Answer: If destination file is not available then it will generate exception :- "source file is not found".

4 Practical:-4 Implement a program to implement simple lexical analyzer.

Program:

```

1  #include <fstream>
2  #include <iostream>
3  #include <stdlib.h>
4  #include <string.h>
5  #include <ctype.h>
6
7  using namespace std;
8
9  bool isPunctuator(char ch) //check if the given character is a
10 punctuator or not
11 {
12     if (ch == ' ' || ch == '+' || ch == '-' || ch == '*' ||
13         ch == '/' || ch == ',' || ch == ';' || ch == '>' ||
14         ch == '<' || ch == '=' || ch == '(' || ch == ')' ||
15         ch == '[' || ch == ']' || ch == '{' || ch == '}' ||
16         ch == '&' || ch == '|')
17     {
18         return true;
19     }
20     return false;
21 }
22
23 bool validIdentifier(char *str) //check if the given identifier is valid or not
24 {
25     if (str[0] == '0' || str[0] == '1' || str[0] == '2' ||
26         str[0] == '3' || str[0] == '4' || str[0] == '5' ||
27         str[0] == '6' || str[0] == '7' || str[0] == '8' ||
28         str[0] == '9' || isPunctuator(str[0]) == true)
29     {
30         return false;
31     } //if first character of string is a digit or a special character, identifier is not valid
32     int i, len = strlen(str);
33     if (len == 1)
34     {
35         return true;
36     } //if length is one, validation is already completed, hence return true
37     else
38     {
39         for (i = 1; i < len; i++) //identifier cannot contain special characters
40         {
41             if (isPunctuator(str[i]) == true)
42             {
43                 return false;
44             }
45         }
46     }
47     return true;
48 }
49
50 bool isOperator(char ch) //check if the given character is an operator or not
51 {
52     if (ch == '+' || ch == '-' || ch == '*' ||

```

```

53     ch == '/' || ch == '>' || ch == '<' ||
54     ch == '=' || ch == '|' || ch == '&')
55 {
56     return true;
57 }
58 return false;
59 }
60
61 bool isKeyword(char *str) //check if the given substring is a keyword or not
62 {
63     if (!strcmp(str, "if") || !strcmp(str, "else") || !strcmp(str, "while") || !strcmp(str, "do") ||
64         !strcmp(str, "break") || !strcmp(str, "continue") || !strcmp(str, "int") || !strcmp(str, "double") ||
65         !strcmp(str, "float") || !strcmp(str, "return") || !strcmp(str, "char") || !strcmp(str, "case") ||
66         !strcmp(str, "long") || !strcmp(str, "short") || !strcmp(str, "typedef") || !strcmp(str, "switch") ||
67         !strcmp(str, "unsigned") || !strcmp(str, "void") || !strcmp(str, "static") || !strcmp(str, "struct") ||
68         !strcmp(str, "sizeof") || !strcmp(str, "long") || !strcmp(str, "volatile") || !strcmp(str, "typedef") ||
69         !strcmp(str, "enum") || !strcmp(str, "const") || !strcmp(str, "union") || !strcmp(str, "extern") ||
70         !strcmp(str, "bool"))
71     {
72         return true;
73     }
74     else
75     {
76         return false;
77     }
78 }
79
80 bool isNumber(char *str) //check if the given substring is a number or not
81 {
82     int i, len = strlen(str), numOfDecimal = 0;
83     if (len == 0)
84     {
85         return false;
86     }
87     for (i = 0; i < len; i++)
88     {
89         if (numOfDecimal > 1 && str[i] == '.')
90         {
91             return false;
92         }
93         else if (numOfDecimal <= 1)
94         {
95             numOfDecimal++;
96         }
97         if (str[i] != '0' && str[i] != '1' && str[i] != '2' && str[i] != '3' && str[i] != '4' && str[i] != '5' &&
98             str[i] != '6' && str[i] != '7' && str[i] != '8' && str[i] != '9' || (str[i] == '-' && i > 0))
99         {
100             return false;
101         }
102     }
103     return true;
104 }
105
106 char *subString(string realStr, int l, int r) //extract the required substring from the main string
107 {
108     int i;
109
110     char *str = (char *)malloc(sizeof(char) * (r - l + 2));
111
112     for (i = l; i <= r; i++)
113     {

```

```

114     str[i - 1] = realStr[i];
115     str[r - 1 + 1] = '\0';
116 }
117 return str;
118 }
119
120 void parse(string str) //parse the expression
121 {
122     int left = 0, right = 0;
123     int len = str.length();
124     while (right <= len && left <= right)
125     {
126         if (isPunctuator(str[right]) == false) //if character is a digit or an alphabet
127         {
128             right++;
129         }
130
131         if (isPunctuator(str[right]) == true && left == right) //if character is a punctuator
132         {
133             if (isOperator(str[right]) == true)
134             {
135                 std::cout << str[right] << " -> OPERATOR\n";
136             }
137             right++;
138             left = right;
139         }
140         else if (isPunctuator(str[right]) == true && left != right || (right == len && left != right))
141             //check if parsed substring is a keyword or identifier or number
142         {
143             char *sub = subString(str, left, right - 1); //extract substring
144
145             if (isKeyword(sub) == true)
146             {
147                 cout << sub << " -> KEYWORD\n";
148             }
149             else if (isNumber(sub) == true)
150             {
151                 cout << sub << " -> NUMBER\n";
152             }
153             else if (validIdentifier(sub) == true && isPunctuator(str[right - 1]) == false)
154             {
155                 cout << sub << " -> VALID IDENTIFIER\n";
156             }
157             else if (validIdentifier(sub) == false && isPunctuator(str[right - 1]) == false)
158             {
159                 cout << sub << " -> NOT A VALID IDENTIFIER\n";
160             }
161
162             left = right;
163         }
164     }
165     return;
166 }
167
168 int main()
169 {
170     char filename[20];
171     cin >> filename;
172     ifstream fin(filename);
173
174     string str;

```



```

175
176     while (fin)
177     {
178         // Read a Line from File
179         getline(fin, str);
180
181         parse(str);
182         // Print line in Console
183         // cout << str << endl;
184     }
185     return 0;
186 }

```

OUTPUT:

The screenshot shows the Visual Studio Code editor with a C++ file named `code.cpp` open. The code implements a lexical analyzer using regular expressions to identify tokens in the input string `2a + 3b = 6c`. The tokens identified are `NUMBERS`, `IDENTIFIERS`, and `OPERATORS`.

The code in `code.cpp` is as follows:

```

1  #include "bits/stdc++.h"
2  using namespace std;
3
4  int32_t main()
5  {
6      string str;
7      getline(cin, str);
8
9      // define list of patterns
10     map<string, string> patterns{
11         {"[0-9]+", "NUMBERS"},
12         {"[a-z]+", "IDENTIFIERS"},
13         {"\\+|\\-|\\*|\\/+", "OPERATORS"}};
14
15     // storage for results
16     map<size_t, pair<string, string>> matches;
17
18     for (auto pat = patterns.begin(); pat != patterns.end(); ++pat)
19     {
20         regex r(pat->first);
21         auto words_begin = sregex_iterator(str.begin(), str.end(), r);
22         auto words_end = sregex_iterator();
23
24         for (auto it = words_begin; it != words_end; ++it)
25             matches[it->position()] = make_pair(it->str(), pat->second);

```

The terminal output shows the execution of the program:

```

PS D:\CD\Lexical analyzer> cd "d:\CD\Lexical analyzer\" ; if ($?) { g++ code.cpp -o code } ; if ($?) { .\code }
2a + 3b = 6c
2 -> NUMBERS
a -> IDENTIFIERS
+ -> OPERATORS
3 -> NUMBERS
b -> IDENTIFIERS
6 -> NUMBERS
c -> IDENTIFIERS
PS D:\CD\Lexical analyzer>

```

Answer the following Questions based on Program.

1. What is the main function of Lexical Analyzer?

Answer: The main task of lexical analysis is to read input characters in the code and produce tokens. "Get next token" is a command which is sent from the parser to the lexical analyzer. On receiving this command, the lexical analyzer scans the input until it finds the next token.

2. How many tokens are there in following code?

printf ("i = %d, &i = %x", i, &i);

Answer: identifier: 2, character: 4, alphabet: 2,
total = 8 tokens.

3. List the secondary duties of Lexical Analyzer?

Answer: The lexical analyzer is the first phase of compiler. Its main task is to read the input characters and produce as output a sequence of tokens that the parser uses for syntax analysis.

- It is implemented by making lexical analyzer be a subroutine.
- Upon receiving a "get next token" command from parser, the lexical analyzer reads the input character until it can identify the next token.
- It may also perform secondary task at user interface.
- One such task is stripping out from the source program comments and white space in the form of blanks, tabs, and newline characters.
- Some lexical analyzer are divided into cascade of two phases, the first called scanning and second is "lexical analysis".
- The scanner is responsible for doing simple task while lexical analysis does the more complex task.

5 Program:-5 Write a program to eliminate left recursion from a grammar.

Program:

```

1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main()
6  {
7      int n, j, l, i, k;
8      int length[10] = {};
9      string d, a, b, flag;
10     char c;
11
12     cout << "Enter Parent Non-Terminal: ";
13     cin >> c;
14     d.push_back(c);
15     a += d + "'->";
16     d += "->";
17     b += d;
18     cout << "Enter Total of productions: ";
19     cin >> n;
20
21     for (int i = 0; i < n; i++)
22     {
23         cout << "Enter Production ";
24         cout << i + 1 << " : ";
25         cin >> flag;
26         length[i] = flag.size();
27         d += flag;
28         if (i != n - 1)
29         {
30             d += "|";
31         }
32     }
33
34     for (i = 0, k = 3; i < n; i++)
35     {
36         if (d[0] != d[k])
37         {
38             if (d[k] == '#')
39             {
40                 b.push_back(d[0]);
41                 b += "' ";
42             }
43             else
44             {
45                 for (j = k; j < k + length[i]; j++)
46                 {
47                     b.push_back(d[j]);
48                 }
49                 k = j + 1;
50                 b.push_back(d[0]);
51                 b += "'| ";
52             }

```

```
53     }
54     else
55     {
56         if (d[k] != '#')
57         {
58             for (l = k + 1; l < k + length[i]; l++)
59             {
60                 a.push_back(d[l]);
61             }
62             k = l + 1;
63             a.push_back(d[0]);
64             a += "\\|";
65         }
66     }
67 }
68 a += "#";
69 cout << b << endl;
70 cout << a << endl;
71 return 0;
72 }
```

OUTPUT:

CPH JUDGE: RESULTS

Input: B
2
Be b

Expected Output: Enter Parent Non-Terminal: Enter Total of productions: Enter Production 1 :Enter Production 2 :B->bB'| B'->eB'|#

Received Output: Enter Parent Non-Terminal: Enter Total of productions: Enter Production 1 :Enter Production 2 :Enter Production 3 :A->aA'| A'->BdA'|aA'|#

2 Failed 261ms

+ New Testcase

Set ONLINE_JUDGE

Run All + New

Stop

code.cpp

```

Left recursion > C++ code.cpp > main()
1 #include <iostream>
2 #include <string>
3 using namespace std;
4 int main()
5 {
6     int n, j, l, i, k;
7     int length[10] = {};
8     string d, a, b, flag;
9     char c;
10
11     cout << "Enter Parent Non-Terminal: ";
12     cin >> c;
13
14     d.push_back(c);
15     a += d + "\'->";
16     d += "->";
17     b += d;
18
19     cout << "Enter Total of productions: ";
20     cin >> n;
21
22     for (int i = 0; i < n; i++)
23     {
24         cout << "Enter Production ";
25         cout << i + 1 << " : ";
    
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```

Enter Parent Non-Terminal: B
Enter Total of productions: 2
Enter Production 1 :Be b
Enter Production 2 :B->bB'|
B'->eB'|#
PS D:\CD\Left recursion> cd "d:\CD\Left recursion\" ; if ($?) { g++ code.cpp -o code } ; if ($?) { .\code }
Enter Parent Non-Terminal: A
Enter Total of productions: 3
Enter Production 1 :ABd Aa a
Enter Production 2 :Enter Production 3 :A->aA'|
A'->BdA'|aA'|#
PS D:\CD\Left recursion>
    
```

Answer the following Questions based on Program.

1. Why left Recursion creates a problem in Top Down Parsing?

Answer: Left recursion often poses problems for parsers, either because it leads them into infinite recursion (as in the case of most top-down parsers) or because they expect rules in a normal form that forbids it. Therefore, a grammar is often preprocessed to eliminate the left recursion.

2. How many types of Left Recursion do you know? List their Names.

Answer:

- 1) Left Recursion
- 2) Right Recursion
- 3) Indirect Recursion

6 Program:-6 Write a program to left factor a grammar.

Program:

```

1  #include <iostream>
2  #include <string>
3  using namespace std;
4  int main()
5  {
6      string ip, op1, op2, temp;
7      int sizes[10] = {};
8      char c;
9      int n, j, l;
10     cout << "Enter the Parent Non-Terminal : ";
11     cin >> c;
12     ip.push_back(c);
13     op1 += ip + "\\'->";
14     op2 += ip + "\\'\'->";
15     ;
16     ip += "->";
17     cout << "Enter the number of productions : ";
18     cin >> n;
19     for (int i = 0; i < n; i++)
20     {
21         cout << "Enter Production " << i + 1 << " : ";
22         cin >> temp;
23         sizes[i] = temp.size();
24         ip += temp;
25         if (i != n - 1)
26             ip += "|";
27     }
28     cout << "Production Rule : " << ip << endl;
29     char x = ip[3];
30     for (int i = 0, k = 3; i < n; i++)
31     {
32         if (x == ip[k])
33         {
34             if (ip[k + 1] == '|')
35             {
36                 op1 += "#";
37                 ip.insert(k + 1, 1, ip[0]);
38                 ip.insert(k + 2, 1, '\\');
39                 k += 4;
40             }
41             else
42             {
43                 op1 += "|" + ip.substr(k + 1, sizes[i] - 1);
44                 ip.erase(k - 1, sizes[i] + 1);
45             }
46         }
47         else
48         {
49             while (ip[k++] != '|')
50                 ;
51         }
52     }
53     char y = op1[6];
54     for (int i = 0, k = 6; i < n - 1; i++)

```

```
55     {
56         if (y == op1[k])
57         {
58             if (op1[k + 1] == '|')
59             {
60                 op2 += "#";
61                 op1.insert(k + 1, 1, op1[0]);
62                 op1.insert(k + 2, 2, '\\');
63                 k += 5;
64             }
65             else
66             {
67                 temp.clear();
68                 for (int s = k + 1; s < op1.length(); s++)
69                     temp.push_back(op1[s]);
70                 op2 += "|" + temp;
71                 op1.erase(k - 1, temp.length() + 2);
72             }
73         }
74     }
75     op2.erase(op2.size() - 1);
76     cout << "After Left Factoring : " << endl;
77     cout << ip << endl;
78     cout << op1 << endl;
79     cout << op2 << endl;
80     return 0;
81 }
```

OUTPUT:

```

15
16 ip += ">";
17 cout << "Enter the number of productions : ";
18 cin >> n;
19 for (int i = 0; i < n; i++)
20 {
21     cout << "Enter Production " << i + 1 << " : ";
22     cin >> temp;
23     sizes[i] = temp.size();
24     ip += temp;
25     if (i != n - 1)
26         ip += "|";
27 }
28 cout << "Production Rule : " << ip << endl;
29 char x = ip[3];
30 for (int i = 0, k = 3; i < n; i++)
31 {
32     if (x == ip[k])
33     {
34         if (ip[k + 1] == '|')
35         {
36             op1 += "#";
37             ip.insert(k + 1, 1, ip[0]);
38             ip.insert(k + 2, 1, '\\');
39             k += 4;
40         }
41     }
42 }
43 cout << "After Left Factoring : " << ip << endl;
44 
```

PS D:\CD\left factor> cd "d:\CD\left factor\" ; if (\$?) { g++ left_factor.cpp -o left_factor } ; if (\$?) { .\left_factor }
 Enter the Parent Non-Terminal : S
 Enter the number of productions : 3
 Enter Production 1 : iEtS
 Enter Production 2 : iEtSeS
 Enter Production 3 : a
 Production Rule : S->iEtS|iEtSeSa
 After Left Factoring :
 S->a
 S'>->|
 S''>->S|iEtSeS
 PS D:\CD\left factor>

Answer the following Questions based on Program.

1. Why left Factoring creates a problem in Top Down Parsing?

Answer: A top down parser generated from the grammar is not efficient as it requires backtracking.

2. Remove Left Recursion from $A \rightarrow Xyz \mid Xz$

Answer: $A \rightarrow XA'$

$A' \rightarrow yz \mid z$

7 Program:-7 Write a program that will find the FIRST SET of the grammar.

Program:

```

1  #include "bits/stdc++.h"
2  using namespace std;
3
4  void followfirst(char, int, int);
5  void follow(char c);
6  void findfirst(char, int, int);
7
8  int cnt, n = 0;
9  char calc_first[10][100];
10 char calc_follow[10][100];
11 int m = 0;
12 char production[10][10];
13 char f[10], first[10];
14 int k;
15 char ck;
16 int e;
17
18 void init_code()
19 {
20     #ifndef ONLINE_JUDGE
21         freopen("input.txt", "r", stdin);
22         freopen("output.txt", "w", stdout);
23     #endif
24 }
25
26 int32_t main()
27 {
28     int jm = 0;
29     int km = 0;
30     int i, choice, cnt = 0;
31     char c, ch;
32     init_code();
33     string str;
34
35     while (cin)
36     {
37         // Read a Line from File
38         getline(cin, str);
39
40         strcpy(production[cnt++], str.c_str());
41     }
42
43     int kay;
44     char done[cnt];
45     int ptr = -1;
46     for (k = 0; k < cnt; k++)
47     {
48         for (kay = 0; kay < 100; kay++)
49         {
50             calc_first[k][kay] = '!';
51         }
52     }

```

```

53     int point1 = 0, point2, xxx;
54
55     for (k = 0; k < cnt; k++)
56     {
57         c = production[k][0];
58         point2 = 0;
59         xxx = 0;
60         for (kay = 0; kay <= ptr; kay++)
61             if (c == done[kay])
62                 xxx = 1;
63
64         if (xxx == 1)
65             continue;
66         findfirst(c, 0, 0);
67         ptr += 1;
68         done[ptr] = c;
69         cout << "\n First(" << c << ") = { ";
70         calc_first[point1][point2++] = c;
71
72         for (i = 0 + jm; i < n; i++)
73         {
74             int lark = 0, chk = 0;
75             for (lark = 0; lark < point2; lark++)
76             {
77                 if (first[i] == calc_first[point1][lark])
78                 {
79                     chk = 1;
80                     break;
81                 }
82             }
83             if (chk == 0)
84             {
85                 cout << first[i] << ", ";
86                 calc_first[point1][point2++] = first[i];
87             }
88         }
89         cout << "]\n";
90         jm = n;
91         point1++;
92     }
93 }
94
95 void findfirst(char c, int q1, int q2)
96 {
97     int j;
98     if (!isupper(c))
99     {
100         first[n++] = c;
101     }
102     for (j = 0; j < cnt; j++)
103     {
104         if (production[j][0] == c)
105         {
106             if (production[j][2] == '#')
107             {
108                 if (production[q1][q2] == '\0')
109                     first[n++] = '#';
110                 else if (production[q1][q2] != '\0' && (q1 != 0 || q2 != 0))
111                 {
112                     findfirst(production[q1][q2], q1, (q2 + 1));
113                 }

```

```
114         else
115             first[n++] = '#';
116     }
117     else if (!isupper(production[j][2]))
118     {
119         first[n++] = production[j][2];
120     }
121     else
122     {
123         findfirst(production[j][2], j, 3);
124     }
125 }
126 }
127 }
```

OUTPUT:

The screenshot shows a C++ program in VS Code. The main window displays the code for `first_follow.cpp`. The code implements a function `followfirst(char, int, int)` that processes input from `input.txt` and writes the output to `output.txt`. The code includes logic to calculate the first set for each non-terminal and to generate the follow set for each terminal.

```

72 done[ptr] = c;
73 cout << "\n First(" << c << " ) = { ";
74 calc_first[point1][point2++] = c;
75
76 for (i = 0 + jm; i < n; i++)
77 {
78     int lark = 0, chk = 0;
79     for (lark = 0; lark < point2; lark++)
80     {
81         if (first[i] == calc_first[point1][lark])
82         {
83             chk = 1;
84             break;
85         }
86     }
87     if (chk == 0)
88     {
89         cout << first[i] << " , ";
90         calc_first[point1][point2++] = first[i];
91     }
92 }
93 cout << " }\n";
94 jm = n;
95 point1++;
96 cout << "\n";
97
98 char donee[cnt];
99 ptr = -1;
100
101 for (k = 0; k < cnt; k++)
102 {
103     for (kay = 0; kay < 100; kay++)
104     {
105         calc_follow[k][kay] = ' ';
106     }
107 }
108
109 }

```

The `input.txt` file contains the following input:

```

1 S->Aa
2 S->Ac
3 A->b

```

The `output.txt` file contains the following output:

```

1 First(S) = { }
2
3 First(A) = { [ ] }

```

Answer the following Questions based on Program.

1. Write Rules for finding FIRST.

Answer:

- If X is a terminal then $\text{First}(X)$ is just X !
- If there is a Production $X \rightarrow \text{...}$ then add to $\text{First}(X)$
- If there is a Production $X \rightarrow Y_1Y_2..Y_k$ then add $\text{first}(Y_1Y_2..Y_k)$ to $\text{first}(X)$
- $\text{First}(Y_1Y_2..Y_k)$ is either
 - $\text{First}(Y_1)$ (if $\text{First}(Y_1)$ doesn't contain)
 - OR (if $\text{First}(Y_1)$ does contain) then $\text{First}(Y_1Y_2..Y_k)$ is everything in $\text{First}(Y_1)$ (except for) as well as everything in $\text{First}(Y_2..Y_k)$
 - If $\text{First}(Y_1) \text{First}(Y_2).. \text{First}(Y_k)$ all contain then add to $\text{First}(Y_1Y_2..Y_k)$ as well.

8 Program:-8 Write a program that will find the FOLLOW SET of the grammar.

Program:

```

1  #include "bits/stdc++.h"
2  using namespace std;
3
4  void followfirst(char, int, int);
5  void follow(char c);
6  void findfirst(char, int, int);
7
8  int cnt, n = 0;
9  char calc_first[10][100];
10 char calc_follow[10][100];
11 int m = 0;
12 char production[10][10];
13 char f[10], first[10];
14 int k;
15 char ck;
16 int e;
17
18 void init_code()
19 {
20     #ifndef ONLINE_JUDGE
21         freopen("input.txt", "r", stdin);
22         freopen("output.txt", "w", stdout);
23     #endif
24 }
25
26 int32_t main()
27 {
28     int jm = 0;
29     int km = 0;
30     int i, choice, cnt = 0;
31     char c, ch;
32     init_code();
33     string str;
34
35     while (cin)
36     {
37         // Read a Line from File
38         getline(cin, str);
39
40         strcpy(production[cnt++], str.c_str());
41     }
42
43     int kay;
44     char done[cnt];
45     int ptr = -1;
46     for (k = 0; k < cnt; k++)
47     {
48         for (kay = 0; kay < 100; kay++)
49         {
50             calc_first[k][kay] = '!';
51         }
52     }

```

```

53     int point1 = 0, point2, xxx;
54
55     for (k = 0; k < cnt; k++)
56     {
57         c = production[k][0];
58         point2 = 0;
59         xxx = 0;
60         for (kay = 0; kay <= ptr; kay++)
61             if (c == done[kay])
62                 xxx = 1;
63
64         if (xxx == 1)
65             continue;
66         findfirst(c, 0, 0);
67         ptr += 1;
68         done[ptr] = c;
69         cout << "\n First(" << c << ") = { ";
70         calc_first[point1][point2++] = c;
71
72         for (i = 0 + jm; i < n; i++)
73         {
74             int lark = 0, chk = 0;
75             for (lark = 0; lark < point2; lark++)
76             {
77                 if (first[i] == calc_first[point1][lark])
78                 {
79                     chk = 1;
80                     break;
81                 }
82             }
83             if (chk == 0)
84             {
85                 cout << first[i] << ", ";
86                 calc_first[point1][point2++] = first[i];
87             }
88         }
89         cout << "]\n";
90         jm = n;
91         point1++;
92     }
93     cout << "\n";
94
95     char donee[cnt];
96     ptr = -1;
97     for (k = 0; k < cnt; k++)
98     {
99         for (kay = 0; kay < 100; kay++)
100         {
101             calc_follow[k][kay] = '!';
102         }
103     }
104     point1 = 0;
105     int land = 0;
106     for (e = 0; e < cnt; e++)
107     {
108         ck = production[e][0];
109         point2 = 0;
110         xxx = 0;
111         for (kay = 0; kay <= ptr; kay++)
112             if (ck == donee[kay])
113                 xxx = 1;

```

```

114
115     if (xxx == 1)
116         continue;
117     land += 1;
118     follow(ck);
119     ptr += 1;
120     donee[ptr] = ck;
121     cout << " Follow(" << ck << ") = { ";
122     calc_follow[point1][point2++] = ck;
123
124     for (i = 0 + km; i < m; i++)
125     {
126         int lark = 0, chk = 0;
127         for (lark = 0; lark < point2; lark++)
128         {
129             if (f[i] == calc_follow[point1][lark])
130             {
131                 chk = 1;
132                 break;
133             }
134         }
135         if (chk == 0)
136         {
137             cout << f[i] << ", ";
138             calc_follow[point1][point2++] = f[i];
139         }
140     }
141     cout << " }\n\n";
142     km = m;
143     point1++;
144 }
145 }
146
147 void follow(char c)
148 {
149     int i, j;
150     if (production[0][0] == c)
151     {
152         f[m++] = '$';
153     }
154     for (i = 0; i < 10; i++)
155     {
156         for (j = 2; j < 10; j++)
157         {
158             if (production[i][j] == c)
159             {
160                 if (production[i][j + 1] != '\0')
161                 {
162                     followfirst(production[i][j + 1], i, (j + 2));
163                 }
164                 if (production[i][j + 1] == '\0' && c != production[i][0])
165                 {
166                     follow(production[i][0]);
167                 }
168             }
169         }
170     }
171 }
172
173 void followfirst(char c, int c1, int c2)
174 {

```

```
175     int k;
176     if (!(isupper(c)))
177         f[m++] = c;
178     else
179     {
180         int i = 0, j = 1;
181         for (i = 0; i < cnt; i++)
182         {
183             if (calc_first[i][0] == c)
184                 break;
185         }
186         while (calc_first[i][j] != '!')
187         {
188             if (calc_first[i][j] != '#')
189             {
190                 f[m++] = calc_first[i][j];
191             }
192             else
193             {
194                 if (production[c1][c2] == '\\0')
195                 {
196                     follow(production[c1][0]);
197                 }
198                 else
199                 {
200                     followfirst(production[c1][c2], c1, c2 + 1);
201                 }
202             }
203             j++;
204         }
205     }
206 }
```

OUTPUT:

```

first_follow > C++ first_follow.cpp > main()
118     for (kay = 0; kay <= ptr; kay++)
119     {
120         if (ck == donee[kay])
121             xxx = 1;
122     }
123     if (xxx == 1)
124         continue;
125     land += 1;
126     follow(ck);
127     ptr += 1;
128 }
129 donee[ptr] = ck;
130 cout << " Follow(" << ck << ") = { ";
131 calc_follow[point1][point2++] = ck;
132 }
133 for (i = 0 + km; i < m; i++)
134 {
135     int lark = 0, chk = 0;
136     for (lark = 0; lark < point2; lark++)
137     {
138         if (f[i] == calc_follow[point1][lark])
139         {
140             chk = 1;
141             break;
142         }
143     }
144     if (chk == 0)
145     {
146         cout << f[i] << ", ";
147         calc_follow[point1][point2++] = f[i];
148     }
149 }
150 cout << " }\n\n";
151 km = m;
152 point1++;
153 }
154 }

```

```

first_follow > input.txt
1 S->Aa
2 S->Ac
3 A->b

```

```

first_follow > output.txt
1
2 First(S) = { }
3
4 First(A) = { }
5
6 Follow(S) = { $, }
7
8 Follow(A) = { a, c, }
9
10

```

Answer the following Questions based on Program.

1. Write Rules for finding FOLLOW.

Answer:

- First put \$ (the end of input marker) in Follow(S) (S is the start symbol)
- If there is a production $A \rightarrow aBb$, (where a can be a whole string) then everything in FIRST(b) except for is placed in FOLLOW(B).
- If there is a production $A \rightarrow aB$, then everything in FOLLOW(A) is in FOLLOW(B)
- If there is a production $A \rightarrow aBb$, where FIRST(b) contains , then everything in FOLLOW(A) is in FOLLOW(B)

9 Program:-9 Implement RDP for a grammar.

Program:

```

1  #include "stdio.h"
2  #include "conio.h"
3  #include "string.h"
4  #include "stdlib.h"
5  #include "ctype.h"
6
7  char ip_sym[15], ip_ptr = 0, op[50], tmp[50];
8  void e_prime();
9  void e();
10 void t_prime();
11 void t();
12 void f();
13 void advance();
14 int n = 0;
15 void e()
16 {
17     strcpy(op, "TE'");
18     printf("E=%-25s", op);
19     printf("E->TE'\n");
20     t();
21     e_prime();
22 }
23
24 void e_prime()
25 {
26     int i, n = 0, l;
27     for (i = 0; i <= strlen(op); i++)
28         if (op[i] != 'e')
29             tmp[n++] = op[i];
30     strcpy(op, tmp);
31     l = strlen(op);
32     for (n = 0; n < l && op[n] != 'E'; n++)
33         ;
34     if (ip_sym[ip_ptr] == '+')
35     {
36         i = n + 2;
37         do
38         {
39             op[i + 2] = op[i];
40             i++;
41         } while (i <= l);
42         op[n++] = '+';
43         op[n++] = 'T';
44         op[n++] = 'E';
45         op[n++] = '\0';
46         printf("E=%-25s", op);
47         printf("E'->+TE'\n");
48         advance();
49         t();
50         e_prime();
51     }
52     else
53     {
54         op[n] = 'e';

```

```

55         for (i = n + 1; i <= strlen(op); i++)
56             op[i] = op[i + 1];
57         printf("E=%-25s", op);
58         printf("E'->e");
59     }
60 }
61 void t()
62 {
63     int i, n = 0, l;
64     for (i = 0; i <= strlen(op); i++)
65         if (op[i] != 'e')
66             tmp[n++] = op[i];
67     strcpy(op, tmp);
68     l = strlen(op);
69     for (n = 0; n < l && op[n] != 'T'; n++)
70         ;
71     i = n + 1;
72     do
73     {
74         op[i + 2] = op[i];
75         i++;
76     } while (i < l);
77     op[n++] = 'F';
78     op[n++] = 'T';
79     op[n++] = 39;
80     printf("E=%-25s", op);
81     printf("T->FT'\n");
82     f();
83     t_prime();
84 }
85
86 void t_prime()
87 {
88     int i, n = 0, l;
89     for (i = 0; i <= strlen(op); i++)
90         if (op[i] != 'e')
91             tmp[n++] = op[i];
92     strcpy(op, tmp);
93     l = strlen(op);
94     for (n = 0; n < l && op[n] != 'T'; n++)
95         ;
96     if (ip_sym[ip_ptr] == '*')
97     {
98         i = n + 2;
99         do
100         {
101             op[i + 2] = op[i];
102             i++;
103         } while (i < l);
104         op[n++] = '*';
105         op[n++] = 'F';
106         op[n++] = 'T';
107         op[n++] = 39;
108         printf("E=%-25s", op);
109         printf("T'->*FT'\n");
110         advance();
111         f();
112         t_prime();
113     }
114     else
115     {

```

```

116         op[n] = 'e';
117         for (i = n + 1; i <= strlen(op); i++)
118             op[i] = op[i + 1];
119         printf("E=%-25s", op);
120         printf("T'->e\n");
121     }
122 }
123
124 void f()
125 {
126     int i, n = 0, l;
127     for (i = 0; i <= strlen(op); i++)
128         if (op[i] != 'e')
129             tmp[n++] = op[i];
130     strcpy(op, tmp);
131     l = strlen(op);
132     for (n = 0; n < l && op[n] != 'F'; n++)
133         ;
134     if ((ip_sym[ip_ptr] == 'i') || (ip_sym[ip_ptr] == 'I'))
135     {
136         op[n] = 'i';
137         printf("E=%-25s", op);
138         printf("F->i\n");
139         advance();
140     }
141     else
142     {
143         if (ip_sym[ip_ptr] == '(')
144         {
145             advance();
146             e();
147             if (ip_sym[ip_ptr] == ')')
148             {
149                 advance();
150                 i = n + 2;
151                 do
152                 {
153                     op[i + 2] = op[i];
154                     i++;
155                 } while (i <= l);
156                 op[n++] = '(';
157                 op[n++] = 'E';
158                 op[n++] = ')';
159                 printf("E=%-25s", op);
160                 printf("F->(E)\n");
161             }
162         }
163         else
164         {
165             printf("\n\t syntax error");
166             getch();
167             exit(1);
168         }
169     }
170 }
171
172 void advance()
173 {
174     ip_ptr++;
175 }
176

```

```
177 int main()
178 {
179     int i;
180     printf("\nGrammar without left recursion");
181     printf("\n\t\t E->TE' \n\t\t E'->+TE'|e \n\t\t T->FT' ");
182     printf("\n\t\t T'->*FT'|e \n\t\t F->(E)|i");
183     printf("\n Enter the input expression:");
184     gets(ip_sym);
185     printf("Expressions");
186     printf("\t Sequence of production rules\n");
187     e();
188     for (i = 0; i < strlen(ip_sym); i++)
189     {
190         if (ip_sym[i] != '+' && ip_sym[i] != '*' && ip_sym[i] != '(' && ip_sym[i] != ')' && ip_sym[i] != 'i' && ip_sym[i] != 'e')
191         {
192             printf("\nSyntax error");
193             break;
194         }
195         for (i = 0; i <= strlen(op); i++)
196             if (op[i] != 'e')
197                 tmp[n++] = op[i];
198         strcpy(op, tmp);
199         printf("\nE=%-25s", op);
200     }
201     getch();
202     return 0;
203 }
```

OUTPUT:

```

RDP > C++ rdp.cpp > advance()
88     int i, n = 0, l;
89     for (i = 0; i <= strlen(op); i++)
90         if (op[i] != 'e')
91             tmp[n++] = op[i];
92     strcpy(op, tmp);
93     l = strlen(op);
94     for (n = 0; n < l && op[n] != 'T'; n++)
95         ;
96     if (ip_sym[ip_ptr] == '*')
97     {
98         i = n + 2;
99         do
100         {
101             op[i + 2] = op[i];
102             i++;
103         } while (i < l);
104         op[n++] = '*';
105         op[n++] = 'F';
106         op[n++] = 'T';

```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

Grammar without left recursion

Enter the input expression: i+i*i

Expressions	Sequence of production rules
E->TE'	E->TE'
E->i+TE'	E'->+TE'
E->i+FT'E'	T->FT'
E->i+iT'E'	F->i
E->i+i*FT'E'	T'->*FT'
E->i+i*IT'E'	F->i
E->i+i*ieE'	T'->e
E->i+i*ie	E'->e
E->i+i*i	

Answer the following Questions based on Program.

1. For Top Down Parsing, When We Announce out result for Input String?

Answer: If $X = a = \$$, The parser halts and announces successful completion of parsing. So, when our stack contain $\$$ as next symbol then we can announce out result for input string.

2. When String is accepted, What the Stack Contains?

Answer: When String is accepted, Stack Contains $\$$.

10 Program:-10 To Study about Lexical Analyzer Generator(LEX) and Flex(Fast Lexical Analyzer)

Lex:

```

1  - Lex is a program that generates lexical analyzer. It is used with YACC parser generator.
2  - The lexical analyzer is a program that transforms an input stream into a sequence of tokens.
3  - It reads the input stream and produces the source code as output through implementing the lexical analyzer in the
4  C program.
5
6  The function of Lex is as follows:
7  - Firstly lexical analyzer creates a program lex.l in the Lex language. Then Lex compiler runs the lex.l program and
8  produces a C program lex.yy.c.
9  - Finally C compiler runs the lex.yy.c program and produces an object program a.out.
10 - a.out is lexical analyzer that transforms an input stream into a sequence of tokens.
11
12 Lex file format
13 A Lex program is separated into three sections by %% delimiters. The format of Lex source is as follows:
14
15 { definitions }
16 %%
17 { rules }
18 %%
19 { user subroutines }
20
21 Definitions include declarations of constant, variable and regular definitions.
22
23 Rules define the statement of form p1 {action1} p2 {action2}....pn {action}.
24
25 Where pi describes the regular expression and action1 describes the actions what action the lexical analyzer should
26 take when pattern pi matches a lexeme.
27
28 User subroutines are auxiliary procedures needed by the actions. The subroutine can be loaded with the lexical analyzer
29 and compiled separately.

```

Flex:

```

1  FLEX (fast lexical analyzer generator) is a tool/computer program for generating lexical analyzers.
2  Bison produces parser from the input file provided by the user. The function yylex() is automatically generated by the
3  flex when it is provided with a .l file and this yylex() function is expected by parser to call to retrieve tokens from
4  current/this token stream.
5
6  Program Structure:
7
8  In the input file, there are 3 sections:
9  1. Definition Section: The definition section contains the declaration of variables, regular definitions, manifest
10 constants. In the definition section, text is enclosed in
11 %{ %} brackets. Anything written in this brackets is copied directly to the file lex.yy.c
12
13 Syntax:
14 %{

```

```

15      // Definitions
16  %{
17
18  2. Rules Section: The rules section contains a series of rules in the form: pattern action and pattern must be
19  unintended and action begin on the same line in {} brackets. The rule section is enclosed in %% %.
20
21  Syntax:
22  %%
23  pattern action
24  %%
25
26  3. User Code Section: This section contains C statements and additional functions.
27  We can also compile these functions separately and load with the lexical analyzer.
28
29  Basic Program Structure:
30  %{
31      // Definitions
32  %{
33  %%
34  Rules
35  %%
36
37  How to run the program:
38  To run the program, it should be first saved with the extension .l or .lex. Run the below commands on terminal in order
39  to run the program file.
40
41  Step 1: lex filename.l or lex filename.lex depending on the extension file is saved with
42  Step 2: gcc lex.yy.c
43  Step 3: ./a.out
44  Step 4: Provide the input to program in case it is required

```

Answer the following Questions based on Program.

1. Write Lex Rule for Number.

Answer: digit[0-9]

2. Write Lex Rule for Identifiers.

Answer: {letter}({letter} || {digit}) *

11 Program:-11 A lexer to print out all numbers from a given file. (Hint: By default lex reads from standard input).

Program:

```

1  %{
2  #include <stdio.h>
3  %}
4  %%
5  [0-9]* {printf("%s",yytext);}
6  . ;
7  /n ;
8  %%
9  int main(void)
10 {
11 yylex();
12 printf("\n");
13 return 0;
14 }
15 int yywrap()
16 {return(1);}

```

OUTPUT:

```

EXPLORER
├── .vscode
├── cp
├── dir
├── first follow
├── Images
├── Lab manual- Overleaf Format- ...
├── Left recursion
├── lex_yacc
│   ├── a.exe
│   ├── cal.l
│   ├── lex.yy.c
│   ├── lex2.l
│   └── lex3.l
├── Lexical analyzer
├── mkdir
├── 180470107035_CD_Dishen_Ma...
└── Lab manual- Overleaf Format- ...

cal.l
1  %{
2
3  #include<stdio.h>
4
5  %}
6
7  %%
8
9  [0-9]* {printf("%s",yytext);}
10
11 . ;
12
13 \n ;
14
15 %%
16
17 int main(void)
18 {
19
20
21 yylex();
22
23 printf("\n");
24
25 return 0;
26
27 }

TERMINAL
collect2.exe: error: ld returned 1 exit status
PS D:\CD\lex_yacc> g++ lex.yy.c
PS D:\CD\lex_yacc> ./a.exe
johndoe@123
123
ishankishan@9856
9856
sunshine
good$%^&

```

Answer the following Questions based on Program.

1. Write Lex Rule to print Alphabets from given String.

Answer: [a - z] [A Z] +

12 Program:-12 A lexer which adds line numbers to the given file and display the same onto the standard output.

Program:

```

1  %{
2  #include<stdio.h>
3  int yylineno;
4  %}a
5  %%
6  ^(.*)/n {printf("%d\t%s",++yylineno,yytext);}
7  . ;
8  /n ;
9  %%
10 int main(void) {
11     yylex();
12     printf("\n");
13     return 0;
14 }
15 int yywrap()
16 {return(1);}

```

OUTPUT:

The screenshot shows a VS Code editor with a file explorer on the left and a terminal at the bottom. The file explorer shows a project named 'lex_yacc' with files 'a.exe', 'cal.l', 'lex.yy.c', 'lex2.l', and 'lex3.l'. The 'lex2.l' file is selected, and its content is displayed in the editor. The terminal shows the execution of the program, which outputs line numbers and the corresponding text from the input file.

```

lex_yacc > lex2.l
1  %{
2
3  #include<stdio.h>
4
5  int yylineno;
6
7  %}
8  %%
9
10 ^(.*)/n {printf("%d\t%s",++yylineno,yytext);}
11 . ;
12 /n ;
13 %%
14
15 int main(void) {
16     yylex();
17     printf("\n");
18     return 0;
19 }
20
21
22
23
24
25
PS D:\CD\lex_yacc> flex lex2.l
PS D:\CD\lex_yacc> g++ lex.yy.c
PS D:\CD\lex_yacc> ./a.exe
hello i am John
1      hello i am John
2      i am study in vvp engineering college
3      i am study in vvp engineering college
4      i am good programmer
5      i am good programmer
6      i am also a gold medalist
7      i am also a gold medalist

```

13 Program:-13 A lexer, which classifies tokens as words, numbers or "other".

Program:

```

1  %{
2  #include <stdio.h>
3  %}
4  %%
5  [a-zA-Z]+ {printf("Word : %s\n",yytext);}
6  [0-9]+ {printf("Number : %s\n",yytext);}
7  [^ /t\n a-zA-Z0-9]+ {printf("Others : %s\n",yytext);}
8  . ;
9  %%
10 int main(void)
11 {
12     yylex();
13     printf("\n");
14     return 0;
15 }
16
17 int yywrap()
18 {return(1);}

```

OUTPUT:

```

lex_yacc > flex lex3.l
1  %{
2  #include<stdio.h>
3  %}
4  %%
5  [a-zA-Z]+ {printf("Word : %s\n",yytext);}
6  [0-9]+ {printf("Number : %s\n",yytext);}
7  [^ /t\n a-zA-Z0-9]+ {printf("Others : %s\n",yytext);}
8  . ;
9  %%
10 int main(void)
11 {
12     yylex();
13     printf("\n");
14     return 0;
15 }
16
17 int yywrap()
18 {return(1);}

```

PROBLEMS 13 OUTPUT TERMINAL DEBUG CONSOLE

```

PS D:\CD\lex_yacc> flex lex3.l
PS D:\CD\lex_yacc> g++ lex.yy.c
PS D:\CD\lex_yacc> ./a.exe
Word : printf
Others : (
Word : i
Others : =
Others : %
Word : d
Others : ,
Others : &

```