COMPILER DESIGN LAB – 04

AKASH KUMAR SINGH

19th August 2024

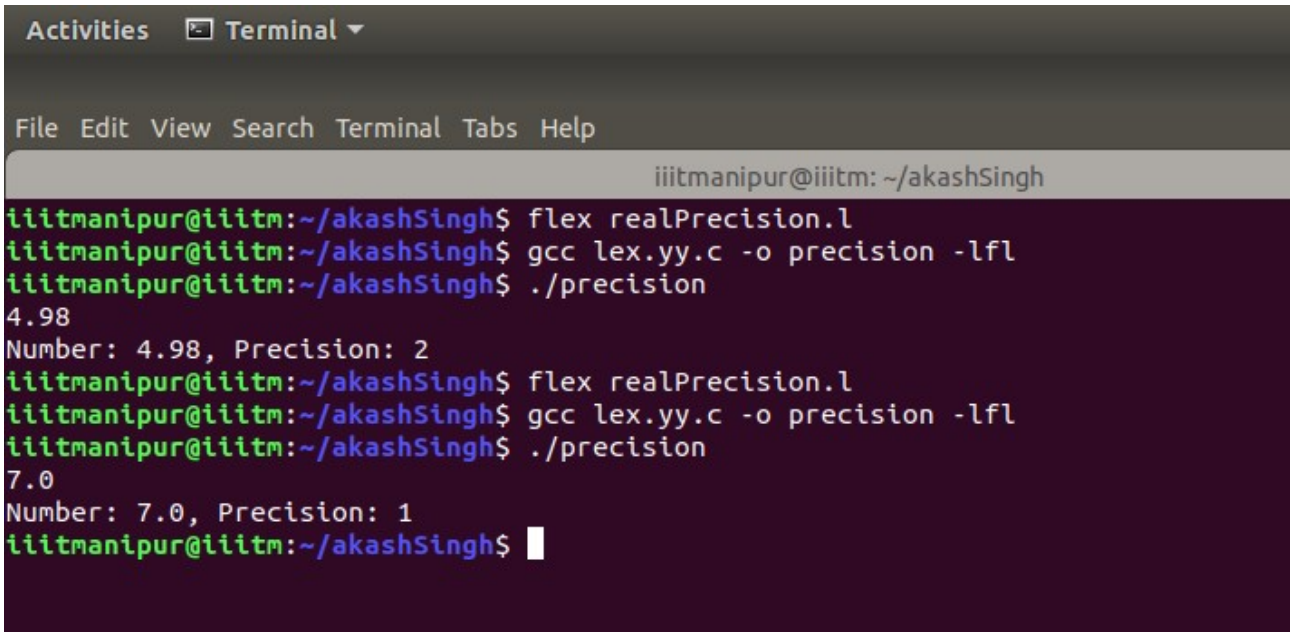
Roll: 220101100

Section: B

Q1. Write a lex program to identify “real precision” of given number.
Code:

```
%{
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void calculate_precision(const char *number_str);
%}

%%
[0-9]*\.[0-9]+  { calculate_precision(yytext); }
[ \t\n]+        ;
.               { }
%%

void calculate_precision(const char *number_str) {
    const char *dot_position = strchr(number_str, '.');

    if (dot_position) {
        int precision = strlen(dot_position + 1);
        printf("Number: %s, Precision: %d\n", number_str, precision);
    } else {
        printf("Number: %s, Precision: 0\n", number_str);
    }
}

int main(int argc, char **argv) {
    yylex();
    return 0;
}
```

Output:



Q2. Write a program to implement the elimination of left recursion.

Code:

```
#include <iostream>
#include <string>
#include <vector>

using namespace std;

void eliminate_left_recursion(const string& production) {
    int pos = production.find("->");
    if (pos == string::npos) {
        cout << "Invalid production format." << endl;
        return;
    }

    string left_part = production.substr(0, pos);
    string right_part = production.substr(pos + 2);

    vector<string> right_parts;
    int start = 0;

    while (true) {
```

```
        int end = right_part.find('|', start);
        if (end == string::npos) {
            right_parts.push_back(right_part.substr(start));
            break;
        }
        right_parts.push_back(right_part.substr(start, end - start));
        start = end + 1;
    }

    vector<string> productions;
    bool has_left_recursion = false;

    for (const string& part : right_parts) {
        if (part.find(left_part) == 0) {
            has_left_recursion = true;
            productions.push_back(left_part + "'->" + part.substr(left_part.length()) +
left_part + "'");
        } else {
            productions.push_back(left_part + "->" + part + left_part + "'");
        }
    }

    if (has_left_recursion) {
        productions.push_back(left_part + "'->ε");
    }

    if (has_left_recursion) {
        cout << "The productions after eliminating Left Recursion are:" << endl;
        for (const string& prod : productions) {
            cout << prod << endl;
        }
    } else {
        cout << "The Given Grammar has no Left Recursion" << endl;
    }
}

int main() {
    string production;
    cout << "Enter the production: ";
    getline(cin, production);
    eliminate_left_recursion(production);
    return 0;
}
```

Output:

```
Activities   Terminal
File  Edit  View  Search  Terminal  Tabs  Help
iiitmanipur@iiitm: ~/akashSingh
iiitmanipur@iiitm:~/akashSingh$ g++ left.cpp
^[[Aiiitmanipur@iiitm:~/akashSingh$ ./a.out
Enter the production: a->a+b|c
The productions after eliminating Left Recursion are:
a'->+ba'
a->ca'
a'->ε
iiitmanipur@iiitm:~/akashSingh$ g++ left.cpp
iiitmanipur@iiitm:~/akashSingh$ ./a.out
Enter the production: a->a+b|a*d|c
The productions after eliminating Left Recursion are:
a'->+ba'
a'->*da'
a->ca'
a'->ε
iiitmanipur@iiitm:~/akashSingh$
```