22CE114

# Internal Practical Exam – CC

- Write program for eliminate left factoring.

CODE:

```cpp
#include <bits/stdc++.h>
using namespace std;

string findCommonPrefix(vector<string> &productions) {
    if (productions.empty()) return "";

    string prefix = productions[0];
    for (string &prod : productions) {
        int len = 0;
        while (len < prefix.size() && len < prod.size() && prefix[len] == prod[len]) {
            len++;
        }
        prefix = prefix.substr(0, len);
        if (prefix.empty()) break;
    }
    return prefix;
}

void removeLeftFactoring(char nonTerminal, vector<string> &productions) {
    string prefix = findCommonPrefix(productions);

    if (prefix.empty() || prefix.size() < 1) {
        cout << nonTerminal << " -> ";
        for (int i = 0; i < productions.size(); i++) {
            if (i > 0) cout << " | ";
            cout << productions[i];
        }
        cout << endl;
        return;
    }
```

22CE114

```
    cout << nonTerminal << " -> " << prefix << nonTerminal << "'" << endl;

    vector<string> newProds;
    for (string &prod : productions) {
        if (prod.size() == prefix.size()) {
            newProds.push_back("^");
        } else {
            newProds.push_back(prod.substr(prefix.size()));
        }
    }

    cout << nonTerminal << "' -> ";
    for (int i = 0; i < newProds.size(); i++) {
        if (i > 0) cout << " | ";
        cout << newProds[i];
    }
    cout << endl;
}

void solve() {
    int n;
    cout << "Enter number of non-terminals: ";
    cin >> n;

    unordered_map<char, vector<string>> grammar;
```

22CE114

```
    unordered_map<char, vector<string>> grammar;

    for (int i = 0; i < n; i++) {
        char nonTerminal;
        int count;
        cout << "Enter non-terminal " << i + 1 << ": ";
        cin >> nonTerminal;

        cout << "Enter number of productions for " << nonTerminal << ": ";
        cin >> count;

        cout << "Enter productions for " << nonTerminal << ":\n";
        for (int j = 0; j < count; j++) {
            string production;
            cin >> production;
            grammar[nonTerminal].push_back(production);
        }
    }

    cout << "\nGrammar after removing left factoring:\n";
    for (auto &entry : grammar) {
        removeLeftFactoring(entry.first, entry.second);
    }
}

int main() {
    solve();
    return 0;
}
```

OUTPUT:

```
Enter number of non-terminals: 1
Enter non-terminal 1: A
Enter number of productions for A: 3
Enter productions for A:
aVF
aBH
aNJ

Grammar after removing left factoring:
A -> aA'
A' -> VF | BH | NJ
```

22CE114

- Write lex program to remove comments from c program.

CODE:

```
%{
#include <stdio.h>
%}

%%
"//".*                          { /* Remove single-line comments */ }
.                               { putchar(yytext[0]); }
\n                              { putchar('\n'); }
%%

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

OUTPUT:

```
c:/mingw/bin/../lib/gcc/mingw32/6.3.0/../../../.
collect2.exe: error: ld returned 1 exit status
PS C:\Users\Deep\Desktop\sem-6\compiler construc
PS C:\Users\Deep\Desktop\sem-6\compiler construc
c:/mingw/bin/../lib/gcc/mingw32/6.3.0/../../../.
c:/mingw/bin/../lib/gcc/mingw32/6.3.0/../../../.
collect2.exe: error: ld returned 1 exit status
```