**COMPILER DESIGN LAB**

**OPERATOR PRECEDENCE PARSER**

**BY-**

**AAKRITI MEHROTRA**

**22BCE1954**

#include <iostream>

#include <map>

#include <stack>

#include <string>

#include <vector>

#include <bits/stdc++.h>

using namespace std;

void displayPrecedenceTable(map<char, map<char, char>> &precedenceTable)

{

cout << "Operator Precedence Table:" << endl;

cout << " ";

for (auto col : precedenceTable)

{

cout << col.first << " ";

}

cout << endl;

for (auto row : precedenceTable)

{

cout << row.first << ": ";

for (auto col : precedenceTable)

{

cout << precedenceTable[row.first][col.first] << " ";

}

cout << endl;

}

}

void readPrecedenceTable(map<char, map<char, char>> &precedenceTable, vector<char> &operators)

{

int n;

cout << "Enter number of operators: ";

cin >> n;

cout << "Enter operators: ";

for (int i = 0; i < n; i++)

{

char op;

cin >> op;

operators.push\_back(op);

}

cout << "Enter the precedence relations: (<, =, >) for each pair of operators:" << endl;

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

{

char relation;

cout << operators[i] << " ? " << operators[j] << ": ";

cin >> relation;

precedenceTable[operators[i]][operators[j]] = relation;

}

}

}

char comparePrecedence(char op1, char op2, map<char, map<char, char>> &precedenceTable)

{

return precedenceTable[op1][op2];

}

bool parseString(string &input, map<char, map<char, char>> &precedenceTable, vector<char> &operators)

{

stack<char> st;

st.push('$');

int i = 0;

while (i < input.length())

{

char curr = input[i];

char top = st.top();

if (find(operators.begin(), operators.end(), curr) == operators.end() && curr != '$')

{

return false;

}

char precedence = comparePrecedence(top, curr, precedenceTable);

if (precedence == '<' || precedence == '=')

{

st.push(curr);

i++;

}

else if (precedence == '>')

{

st.pop();

}

else

{

return false;

}

}

return st.size() == 1 && st.top() == '$';

}

int main()

{

map<char, map<char, char>> precedenceTable;

vector<char> operators;

readPrecedenceTable(precedenceTable, operators);

displayPrecedenceTable(precedenceTable);

string input;

cout << "Enter input string to parse: ";

cin >> input;

input += '$';

if (parseString(input, precedenceTable, operators))

{

cout << "The input string is valid." << endl;

}

else

{

cout << "The input string is invalid." << endl;

}

return 0;

}

OUTPUT:

