**PRINT THE QUADRUPLE**

**BY**

**AAKRITI MEHROTRA**

**22BCE1954**

CODE:

#include <bits/stdc++.h>

using namespace std;

int precedence(char op)

{

if (op == '+' || op == '-')

return 1;

if (op == '\*' || op == '/')

return 2;

return 0;

}

bool isOperator(char c)

{

return (c == '+' || c == '-' || c == '\*' || c == '/');

}

string infixToPostfix(string exp)

{

stack<char> s;

string postfix = "";

for (char c : exp)

{

if (isdigit(c) || isalpha(c))

{

postfix += c;

}

else if (c == '(')

{

s.push(c);

}

else if (c == ')')

{

while (!s.empty() && s.top() != '(')

{

postfix += s.top();

s.pop();

}

s.pop();

}

else if (isOperator(c))

{

while (!s.empty() && precedence(s.top()) >= precedence(c))

{

postfix += s.top();

s.pop();

}

s.push(c);

}

}

while (!s.empty())

{

postfix += s.top();

s.pop();

}

return postfix;

}

void generateIntermediateCode(string postfix, vector<tuple<string, string, string, string>> &quadruples)

{

stack<string> s;

int tempVarCount = 1;

for (char c : postfix)

{

if (isdigit(c) || isalpha(c))

{

string op(1, c);

s.push(op);

}

else if (isOperator(c))

{

string operand2 = s.top();

s.pop();

string operand1 = s.top();

s.pop();

string tempVar = "t" + to\_string(tempVarCount);

cout << tempVar << " = " << operand1 << " " << c << " " << operand2 << endl;

quadruples.push\_back({string(1, c), operand1, operand2, tempVar});

s.push(tempVar);

tempVarCount++;

}

}

}

void printQuadruples(const vector<tuple<string, string, string, string>> &quadruples)

{

cout << "\nQuadruple Representation:" << endl;

cout << "Operator Arg1 Arg2 Result" << endl;

for (const auto &quad : quadruples)

{

cout << get<0>(quad) << " " << get<1>(quad) << " " << get<2>(quad) << " " << get<3>(quad) << endl;

}

}

int main()

{

string infixExp;

cout << "Enter an expression: ";

cin >> infixExp;

string postfixExp = infixToPostfix(infixExp);

cout << "Postfix Expression: " << postfixExp << endl;

vector<tuple<string, string, string, string>> quadruples;

cout << "Intermediate Code:" << endl;

generateIntermediateCode(postfixExp, quadruples);

printQuadruples(quadruples);

return 0;

}

OUTPUT:

