CS3044 LAB – 11

NAME – ROHIT KUMAR
ROLL – 220103032
SEC – B
BRANCH – CSE-AID

Q1. Write a program to implement 3 address code.

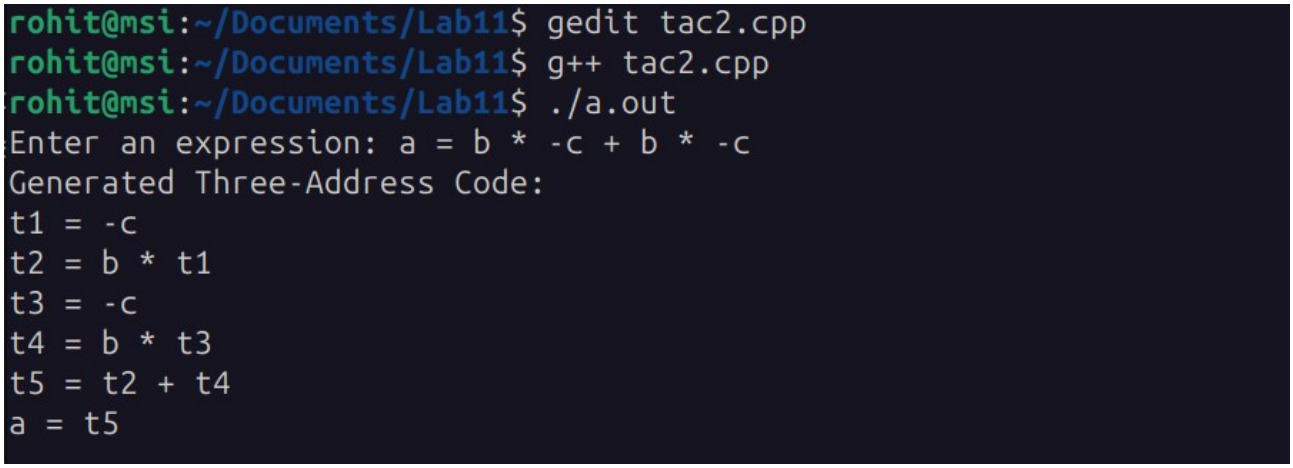
CODE -

```
#include <iostream>
#include <string>
#include <vector>
#include <sstream>
#include <stack>
#include <cctype>
#include <stdexcept>

class ThreeAddressCode {
private:
    std::vector<std::string> code;
    int tempCounter;

    std::string newTemp() {
        return "t" + std::to_string(tempCounter++);
    }

    bool isOperator(char c) {
        return c == '+' || c == '-' || c == '*' || c == '/' || c == '^';
    }

    int precedence(char op) {
        if (op == '^') return 3;
        if (op == '*' || op == '/') return 2;
        if (op == '+' || op == '-') return 1;
        return 0;
    }

    std::string handleUnaryMinus(const std::string& operand) {
        if (operand[0] == '-') {
            std::string temp = newTemp();
            code.push_back(temp + " = -" + operand.substr(1));
            return temp;
        }
        return operand;
    }
```

```
    std::string applyOp(const std::string& a, const std::string& b, char op) {
        std::string temp = newTemp();
        std::string opStr;
        switch (op) {
            case '+': opStr = " + "; break;
            case '-': opStr = " - "; break;
            case '*': opStr = " * "; break;
            case '/': opStr = " / "; break;
            case '^': opStr = " ** "; break;
            default: throw std::runtime_error("Unknown operator");
        }
        code.push_back(temp + " = " + a + opStr + b);
        return temp;
    }

public:
    ThreeAddressCode() : tempCounter(1) {}

    void generate(const std::string& expr) {
        std::istringstream iss(expr);
        std::stack<std::string> values;
        std::stack<char> ops;
        std::string token;

        while (iss >> token) {
            if (token == "=") continue;  // Skip assignment operator

            if (std::isalnum(token[0]) || token[0] == '-') {
                values.push(handleUnaryMinus(token));
            } else if (token == "(") {
                ops.push('(');
            } else if (token == ")") {
                while (!ops.empty() && ops.top() != '(') {
                    std::string b = values.top(); values.pop();
                    std::string a = values.top(); values.pop();
                    char op = ops.top(); ops.pop();
                    values.push(applyOp(a, b, op));
                }
                if (!ops.empty()) ops.pop();
            } else if (isOperator(token[0])) {
                while (!ops.empty() && precedence(ops.top()) >= precedence(token[0])) {
                    std::string b = values.top(); values.pop();
                    std::string a = values.top(); values.pop();
                    char op = ops.top(); ops.pop();
                    values.push(applyOp(a, b, op));
                }
                ops.push(token[0]);
```

```
            }
        }

        while (!ops.empty()) {
            std::string b = values.top(); values.pop();
            std::string a = values.top(); values.pop();
            char op = ops.top(); ops.pop();
            values.push(applyOp(a, b, op));
        }

        if (!values.empty()) {
            code.push_back("a = " + values.top());
        }
    }

    void print() {
        for (const auto& line : code) {
            std::cout << line << std::endl;
        }
    }
};

int main() {
    ThreeAddressCode tac;
    std::string expression;

    while (true) {
        std::cout << "Enter an expression: ";
        std::getline(std::cin, expression);

        if (expression == "quit") {
            break;
        }

        try {
            tac.generate(expression);
            std::cout << "Generated Three-Address Code:" << std::endl;
            tac.print();
        } catch (const std::exception& e) {
            std::cerr << "Error: " << e.what() << std::endl;
        }

        std::cout << std::endl;
    }
}
```

Output -


rohit@msi:~/Documents/Lab11$ gedit tac2.cpp
rohit@msi:~/Documents/Lab11$ g++ tac2.cpp
rohit@msi:~/Documents/Lab11$ ./a.out
Enter an expression: a = b * -c + b * -c
Generated Three-Address Code:
t1 = -c
t2 = b * t1
t3 = -c
t4 = b * t3
t5 = t2 + t4
a = t5