Name: **Manasi Devendra Gursale**
Srn No: **202101135**
Roll No: **24**
Div: **A**

Compiler Design

Assignment 6

Q.Implement constant folding optimization on input 3-address code block in form of quadruples. (In C / C++ / Java).

CODE:

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

using namespace std;

// Define quadruple structure
struct Quadruple {
    char op;
    int arg1;
    int arg2;
    string result;
};

// Function to perform constant folding optimization
vector<Quadruple> constantFolding(vector<Quadruple>& quadruples) {
    vector<Quadruple> newQuadruples;
    for (auto& quad : quadruples) {
        // Check if arg1 is a constant
```

```
        int arg1 = quad.arg1;

        // Check if arg2 is a constant
        int arg2 = quad.arg2;

        // Perform operation based on operator
        if (quad.op == '+') {
            int result = arg1 + arg2;
            newQuadruples.push_back({'=', result, 0, quad.result});
        } else if (quad.op == '-') {
            int result = arg1 - arg2;
            newQuadruples.push_back({'=', result, 0, quad.result});
        } else if (quad.op == '*') {
            int result = arg1 * arg2;
            newQuadruples.push_back({'=', result, 0, quad.result});
        } else if (quad.op == '/') {
            int result = arg1 / arg2;
            newQuadruples.push_back({'=', result, 0, quad.result});
        } else {
            // If operator is not arithmetic, add the quadruple as is
            newQuadruples.push_back(quad);
        }
    }
    return newQuadruples;
}

int main() {
    // Example input 3-address code block in form of quadruples
    vector<Quadruple> quadruples = {
        {'+', 2, 3, "t1"},
        {'*', 5, 4, "t2"},
        {'-', 10, 4, "t3"}
    };
```

```
    // Perform constant folding optimization
    vector<Quadruple> optimizedQuadruples = constantFolding(quadruples);

    // Print optimized quadruples
    for (auto& quad : optimizedQuadruples) {
        cout << quad.op << " " << quad.arg1 << " " << quad.arg2 << " " <<
quad.result << endl;
    }

    return 0;
}
```

OUTPUT: