Name: Khushi Sonkusare
Roll no.: A1_08
PRACTICAL 07

```cpp
#include <iostream>
#include <vector>
#include <unordered_map>
#include <unordered_set>
#include <algorithm>

using namespace std;

struct Instruction {
    int lineNo;
    string code;
};

vector<int> findLeaders(const vector<Instruction>& instructions) {
    vector<int> leaders;
    if (!instructions.empty()) {
        leaders.push_back(instructions[0].lineNo);
    }
    for (size_t i = 0; i < instructions.size(); i++) {
        if (instructions[i].code.find("GOTO") != string::npos) {
            size_t pos = instructions[i].code.find("GOTO");
            int target = stoi(instructions[i].code.substr(pos + 5));
            if (find(leaders.begin(), leaders.end(), target) ==
leaders.end()) {
                leaders.push_back(target);
            }
            if (i + 1 < instructions.size()) {
                leaders.push_back(instructions[i + 1].lineNo);
            }
        }
    }
    sort(leaders.begin(), leaders.end());
    return leaders;
}

unordered_map<int, vector<int>> createBasicBlocks(const
vector<Instruction>& instructions, const vector<int>& leaders) {
    unordered_map<int, vector<int>> basicBlocks;
    int currentBlock = -1;
```

Name: Khushi Sonkusare
Roll no.: A1_08
PRACTICAL 07

```
    for (const auto& instr : instructions) {
        if (find(leaders.begin(), leaders.end(), instr.lineNo) !=
leaders.end()) {
            currentBlock = instr.lineNo;
            basicBlocks[currentBlock] = {};
        }
        basicBlocks[currentBlock].push_back(instr.lineNo);
    }
    return basicBlocks;
}

unordered_map<int, vector<int>> buildCFG(const unordered_map<int,
vector<int>>& basicBlocks) {
    unordered_map<int, vector<int>> cfg;
    for (auto it = basicBlocks.begin(); it != basicBlocks.end(); ++it) {
        int block = it->first;
        const vector<int>& lines = it->second;
        int lastLine = lines.back();
        for (const auto& nextBlock : basicBlocks) {
            if (nextBlock.first > block) {
                cfg[block].push_back(nextBlock.first);
                break;
            }
        }
        if (lastLine != block && find(cfg[block].begin(),
cfg[block].end(), lastLine) == cfg[block].end()) {
            cfg[block].push_back(lastLine);
        }
    }
    return cfg;
}

void printResults(const vector<int>& leaders, const unordered_map<int,
vector<int>>& basicBlocks, const unordered_map<int, vector<int>>& cfg) {
    cout << "Leader Statements:\n";
    for (int leader : leaders) {
        cout << leader << "\n";
    }
    cout << "\nBasic Blocks:\n";
```

Name: Khushi Sonkusare
Roll no.: A1_08
PRACTICAL 07

```
    for (auto it = basicBlocks.begin(); it != basicBlocks.end(); ++it) {
        cout << "Block " << it->first << ": ";
        for (int line : it->second) {
            cout << line << " ";
        }
        cout << "\n";
    }
    cout << "\nProgram Flow Graph (CFG):\n";
    for (auto it = cfg.begin(); it != cfg.end(); ++it) {
        cout << "Block " << it->first << " -> ";
        for (int succ : it->second) {
            cout << succ << " ";
        }
        cout << "\n";
    }
}

int main() {
    vector<Instruction> instructions = {
        {1, "count = 0 L1"},
        {2, "Result = 0"},
        {3, "If count > 20 GOTO 8 L2"},
        {4, "count = count + 1 L3"},
        {5, "increment = 2 * count"},
        {6, "result = result + increment"},
        {7, "GOTO 3"},
        {8, "end L4"}
    };

    vector<int> leaders = findLeaders(instructions);
    unordered_map<int, vector<int>> basicBlocks =
createBasicBlocks(instructions, leaders);
    unordered_map<int, vector<int>> cfg = buildCFG(basicBlocks);

    printResults(leaders, basicBlocks, cfg);
    return 0;
```

Name: Khushi Sonkusare
Roll no.: A1_08
PRACTICAL 07