Практична робота номер 6

```
#include <iostream>

// Структура, що представляє інформацію про студента
struct Student {
    int id;
    std::string name;
    double avarage_mark;

    // Конструктор
    Student(int _id, const std::string& _name, double _avarage_mark)
        : id(_id), name(_name), avarage_mark(_avarage_mark) {}
};

// Вузол дерева
struct Node {
    Student student;
    Node* left;
    Node* right;

    // Конструктор
    Node(const Student& _student) : student(_student), left(nullptr),
right(nullptr) {}
};

// Додавання студента до дерева
Node* insert(Node* root, const Student& student) {
```

```
    if (!root) {
        return new Node(student);
    }

    if (student.avarage_mark >= root->student.avarage_mark) {
        root->right = insert(root->right, student);
    } else {
        root->left = insert(root->left, student);
    }

    return root;
}

//  список студентів в порядку зменшення оцінок
void inOrderTraversal(Node* root) {
    if (!root) return;

    inOrderTraversal(root->right);
    std::cout << "ID: " << root->student.id << ", Name: " << root->student.name << ", avarage_mark: " << root->student.avarage_mark << std::endl;
    inOrderTraversal(root->left);
}

// Функція, яка повертає інформацію про студентів з оцінкою менше, ніж вказана
void findStudentsWithLoweravarage_mark(Node* root, double targetavarage_mark) {
    if (!root) return;
```

```
    if (root->student.avarage_mark < targetavarage_mark) {
        std::cout << "ID: " << root->student.id << ", Name: " << root->student.name << ", avarage_mark: " << root->student.avarage_mark << std::endl;
    }

    if (root->student.avarage_mark >= targetavarage_mark) {
        findStudentsWithLoweravarage_mark(root->right, targetavarage_mark);
    }

    findStudentsWithLoweravarage_mark(root->left, targetavarage_mark);
}

int main() {
    Node* root = nullptr;

    // Додавання студентів до дерева
    root = insert(root, Student(1, "Maria", 5.4));
    root = insert(root, Student(2, "Alina", 9.4));
    root = insert(root, Student(3, "Nikita", 5.2));
    root = insert(root, Student(4, "David", 8.3));
    root = insert(root, Student(5, "Anastasia", 8.8));

    // Обхід дерева та виведення студентів у порядку зменшення оцінок
    std::cout << "Students sorted by decreasing avarage_mark:" << std::endl;
```

```
    inOrderTraversal(root);


    // Знайти студентів з оцінкою менше, ніж 6.5
    double targetavarage_mark = 6.5;
    std::cout << "Students with avarage_mark less than " << targetavarage_mark << ":" << std::endl;
    findStudentsWithLoweravarage_mark(root, targetavarage_mark);


    return 0;
}
```