Отчёт по лабораторной №3. Списки 3.

Задание. Вариант 3. База данных для деканата
Написать программу, осуществляющую работу с базой данных «Деканат» (Группа, фамилия, имя, отчество, оценки за экзамены). Базу данных хранить в памяти в виде массива списков групп. Осуществить добавление, удаление и сортировку элементов списка, формирование списка на отчисление и на начисление стипендии.

Реализация на языке С.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_GROUPS 10
#define NAME_LENGTH 100
#define NUM_GRADES 5
#define PASSING_GRADE 3
#define SCHOLARSHIP_THRESHOLD 4.5

typedef struct Student {
    char surname[NAME_LENGTH];
    char firstname[NAME_LENGTH];
    char patronymic[NAME_LENGTH];
    int grades[NUM_GRADES];
    struct Student *next;
} Student;

typedef struct Group {
    char name[NAME_LENGTH];
    Student *students;
} Group;

// Функция создания нового студента
Student* createStudent(char *surname, char *firstname, char *patronymic, int *grades)
{
    Student *newStudent = (Student *)malloc(sizeof(Student));
    strncpy(newStudent->surname, surname, NAME_LENGTH);
    strncpy(newStudent->firstname, firstname, NAME_LENGTH);
    strncpy(newStudent->patronymic, patronymic, NAME_LENGTH);
    memcpy(newStudent->grades, grades, NUM_GRADES * sizeof(int));
    newStudent->next = NULL;
    return newStudent;
}

// Функция добавления студента в группу
void addStudentToGroup(Group *group, Student *student) {
    student->next = group->students;
    group->students = student;
}

// Функция удаления студента из группы
void removeStudentFromGroup(Group *group, char *surname) {
    Student *current = group->students;
    Student *previous = NULL;

    while (current != NULL && strcmp(current->surname, surname) != 0) {
        previous = current;
```

```
        current = current->next;
    }

    if (current == NULL) {
        printf("Студент с фамилией %s не найден.\n", surname);
        return;
    }

    if (previous == NULL) {
        group->students = current->next;
    } else {
        previous->next = current->next;
    }

    free(current);
}

// Функция для сортировки студентов в группе (по фамилии)
void sortStudentsInGroup(Group *group) {
    if (group->students == NULL) return;

    for (Student *i = group->students; i->next != NULL; i = i->next) {
        for (Student *j = i->next; j != NULL; j = j->next) {
            if (strcmp(i->surname, j->surname) > 0) {
                // Обмен структур
                Student temp = *i;
                *i = *j;
                *j = temp;
                Student *tempNext = i->next;
                i->next = j->next;
                j->next = tempNext;
            }
        }
    }
}

// Функция формирования списка на отчисление
void listForExpulsion(Group *group) {
    Student *current = group->students;
    printf("Список на отчисление в группе %s:\n", group->name);
    while (current != NULL) {
        int failedExams = 0;
        for (int i = 0; i < NUM_GRADES; i++) {
            if (current->grades[i] < PASSING_GRADE) {
                failedExams++;
            }
        }
        if (failedExams > 0) {
            printf("%s %s %s\n", current->surname, current->firstname, current-
>patronymic);
        }
        current = current->next;
    }
}

// Функция формирования списка на начисление стипендии
void listForScholarship(Group *group) {
    Student *current = group->students;
    printf("Список на начисление стипендии в группе %s:\n", group->name);
    while (current != NULL) {
        float average = 0.0;
```

```
        for (int i = 0; i < NUM_GRADES; i++) {
            average += current->grades[i];
        }
        average /= NUM_GRADES;
        if (average >= SCHOLARSHIP_THRESHOLD) {
            printf("%s %s %s\n", current->surname, current->firstname, current-
>patronymic);
        }
        current = current->next;
    }
}

// Функция очистки памяти
void freeGroup(Group *group) {
    Student *current = group->students;
    while (current != NULL) {
        Student *next = current->next;
        free(current);
        current = next;
    }
    group->students = NULL;
}

int main() {
    Group groups[MAX_GROUPS];
    int groupCount = 0;

    // Пример добавления групп и студентов
    strcpy(groups[groupCount].name, "Group1");
    addStudentToGroup(&groups[groupCount], createStudent("Ivanov", "Ivan",
"Ivanovich", (int[]){5, 4, 3, 4, 5}));
    addStudentToGroup(&groups[groupCount], createStudent("Petrov", "Petr",
"Petrovich", (int[]){2, 3, 2, 5, 3}));
    addStudentToGroup(&groups[groupCount], createStudent("Sidorov", "Sidr",
"Sidorovich", (int[]){5, 5, 5, 5, 5}));

    groupCount++;

    // Пример использования функций
    printf("Добавлены начальные данные.\n");
    sortStudentsInGroup(&groups[0]);
    listForExpulsion(&groups[0]);
    listForScholarship(&groups[0]);

    removeStudentFromGroup(&groups[0], "Petrov");

    printf("После удаления:\n");
    sortStudentsInGroup(&groups[0]);
    listForExpulsion(&groups[0]);
    listForScholarship(&groups[0]);

    // Очистка памяти при завершении
    for (int i = 0; i < groupCount; i++) {
        freeGroup(&groups[i]);
    }

    return 0;
}
```

Пример работы программы.

Взаимодействие с программой: