Міністерство освіти і науки України

Харківський національний університет імені В. Н. Каразіна

Факультет комп’ютерних наук

Лабораторна робота №2

з навчальної дисципліни

«Операційні системи»

Виконав:
студент групи КУ-31
Нечипоренко Д.І.

Харків – 2022

Завдання

Задание №0

В пространстве заданы n материальных точек. С некоторого момента точка с наименьшей массой исчезает, передавая свою массу ближайшей к ней точке. Так продолжается до тех пор, пока не останется одна точка. Реализовать этот процесс и найти оставшуюся точку.

Задание №1

Вычислить длину кривой, соответствующую функции $f(x)=\sin(x)^2/(1+\cos(x)^3)$ на отрезке $[a,b]$, приближенно заменив кривую ломаной, полученной в результате разбиения отрезка $[a,b]$ на n равных частей. Элементарные функции $\sin(x)$ и $\cos(x)$ для заданного аргумента x вычислите с заданной точностью ε $(0<\varepsilon<0.1)$ с помощью бесконечных сумм:

$$\sin(x)=x-\frac{x^3}{3!}+\frac{x^5}{5!}-\frac{x^7}{7!}+\ldots=\sum_{n=0}^{\infty}(-1)^n\frac{x^{2n+1}}{(2n+1)!},$$

$$\cos(x)=1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+\ldots=\sum_{n=0}^{\infty}(-1)^n\frac{x^{2n}}{(2n)!}.$$

Считать, что требуемая точность достигнута и все последующие слагаемые можно уже не учитывать, если вычислена сумма нескольких первых слагаемых и очередное слагаемое оказалось по модулю меньше, чем ε.

Замечание: Данную и следующие задачи реализовать в виде многофайловых проектов. Попробовать написать Makefile для каждого проекта. Первую задачу реализовать как «стандартную программу», так и поместив разработанные функции sin и cos в статическую, а затем в динамическую библиотеки.

Задание №2

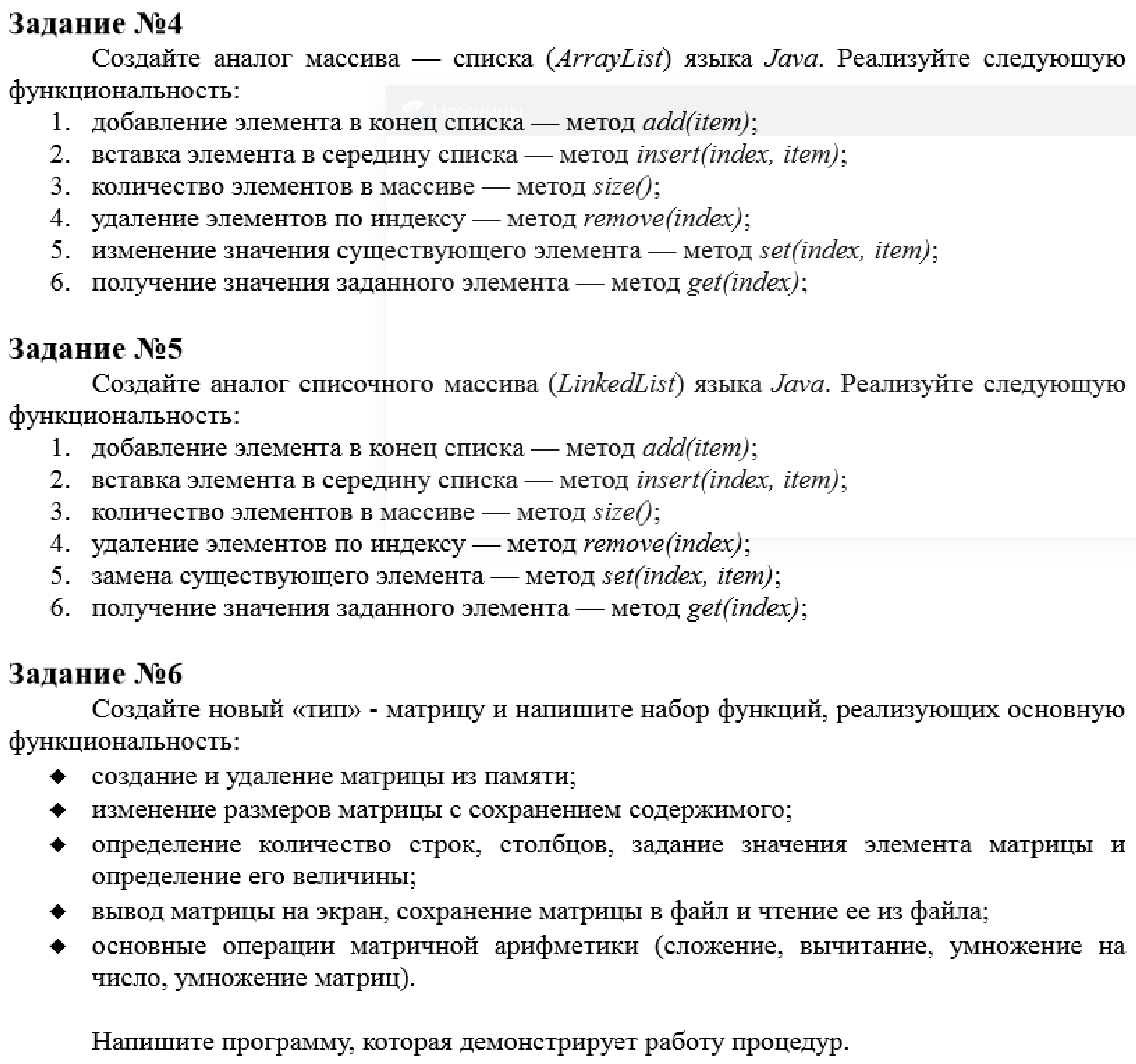
В одномерном массиве, состоящем из n вещественных элементов, вычислить:

- количество элементов, больших среднего значения элементов массива;
- сумму модулей элементов массива, расположенных после первого отрицательного элемента.

Задание №3

В массиве структур содержится информация о зимней сессии 3-го курса. Каждый элемент массива – сведения о конкретном студенте (максимальное количество студентов — 30) – содержит следующие данные: фамилию (до 12 символов), номер группы (от 1 до 4), набранные баллы по трем предметам (*Веб-технологии*, *Укр. язык*, *Проектирование информационных систем*). Напишите программу, которая вводит эту информацию и выводит по запросу такие данные:

- фамилии студентов, имеющих задолженности хотя бы по одному предмету;
- процент студентов, сдавших все экзамены на хорошо и отлично;
- название предмета, который был сдан лучше всего;
- номер группы с наихудшей успеваемостью.



Задание №4

Создайте аналог массива — списка (*ArrayList*) языка *Java*. Реализуйте следующую функциональность:

1. добавление элемента в конец списка — метод *add(item)*;
2. вставка элемента в середину списка — метод *insert(index, item)*;
3. количество элементов в массиве — метод *size()*;
4. удаление элементов по индексу — метод *remove(index)*;
5. изменение значения существующего элемента — метод *set(index, item)*;
6. получение значения заданного элемента — метод *get(index)*;

Задание №5

Создайте аналог списочного массива (*LinkedList*) языка *Java*. Реализуйте следующую функциональность:

1. добавление элемента в конец списка — метод *add(item)*;
2. вставка элемента в середину списка — метод *insert(index, item)*;
3. количество элементов в массиве — метод *size()*;
4. удаление элементов по индексу — метод *remove(index)*;
5. замена существующего элемента — метод *set(index, item)*;
6. получение значения заданного элемента — метод *get(index)*;

Задание №6

Создайте новый «тип» - матрицу и напишите набор функций, реализующих основную функциональность:

- создание и удаление матрицы из памяти;
- изменение размеров матрицы с сохранением содержимого;
- определение количество строк, столбцов, задание значения элемента матрицы и определение его величины;
- вывод матрицы на экран, сохранение матрицы в файл и чтение ее из файла;
- основные операции матричной арифметики (сложение, вычитание, умножение на число, умножение матриц).

Напишите программу, которая демонстрирует работу процедур.

Код:

Task 0

Main

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <math.h>
#include <time.h>
```

```
typedef struct {
    int x;
    int y;
    int weight;
} Point;

Point** generatePoints(int weightFrom, int weightTo, int xFrom, int xTo, int yFrom,
int yTo, int length);
void printPoints(Point** points, int length);
int randomIntNumberInRange(int from, int to);
int indexPointLowestWeight(Point** points, int length);
int findClosestPointIndex(Point** points, int length, int indexPointFrom);
Point**       removePointWithLowestWeightAndSetWeightOfNearestPoint(Point**
points, int length);

int main() {

    srand(time(NULL));

    int length = 5;
    Point** points = generatePoints(1, 10, 1, 10, 1, 10, length);
    printPoints(points, length);
    printf("\n");

    while (length > 1) {
        points                                                              =
removePointWithLowestWeightAndSetWeightOfNearestPoint(points, length);
        length--;

        printPoints(points, length);
        printf("\n");
    }

    printf("LAST POINT INFO\nx: %d, y: %d, weight: %d\n", points[0]->x,
points[0]->y, points[0]->weight);

    return 0;
}



Point** generatePoints(int weightFrom, int weightTo, int xFrom, int xTo, int yFrom,
int yTo, int length){
    Point** points = (Point**) malloc(sizeof(Point*) * length);

    int index;
    for (index = 0; index < length; index++){
        Point* point = (Point*) malloc(sizeof(Point));
        point->x = randomIntNumberInRange(xFrom, xTo);
        point->y = randomIntNumberInRange(yFrom, yTo);
        point->weight = randomIntNumberInRange(weightFrom, weightTo);
        points[index] = point;
    }

    return points;
}

void printPoints(Point** points, int length) {
    int index;
    Point* point;
    for (index = 0; index < length; index++) {
        point = points[index];
        printf("x: %d, y: %d, weight: %d\n", point->x, point->y, point->weight);
    }
}

int randomIntNumberInRange(int from, int to) {
    return rand() % (to - from) + from;
}

Point**      removePointWithLowestWeightAndSetWeightOfNearestPoint(Point**
points, int length) {
    int indexLowerWeightPoint = indexPointLowestWeight(points, length);
    int      closestPointIndex      =      findClosestPointIndex(points,      length,
indexLowerWeightPoint);

    Point* lowerWeightPoint = points[indexLowerWeightPoint];
    Point* closestPoint = points[closestPointIndex];

    closestPoint->weight += lowerWeightPoint->weight;

    free(points[indexLowerWeightPoint]);

    int index;
    for (index = indexLowerWeightPoint; index < length - 1; index++){
        points[index] = points[index + 1];
    }

    points = realloc(points, length);

    return points;
```

```
}

int indexPointLowestWeight(Point** points, int length) {
    int lowestWeightPointIndex = 0;
    int index;
    for (index = 0; index < length; index++){
        if (points[lowestWeightPointIndex]->weight > points[index]->weight){
            lowestWeightPointIndex = index;
        }
    }
    return lowestWeightPointIndex;
}

int findClosestPointIndex(Point** points, int length, int indexPointFrom) {
    Point* pointFrom = points[indexPointFrom];
    int closesPointIndex = 0;
    int minDistance = __INT_MAX__;

    int index;
    for (index = 0; index < length; index++){
        if (index == indexPointFrom){
            continue;
        }

        Point* pointTo = points[index];
        int distance = sqrt(pow(pointTo->x - pointFrom->x, 2) + pow(pointTo->y -
pointFrom->y, 2));

        if (minDistance > distance) {
            minDistance = distance;
            closesPointIndex = index;
        }
    }

    return closesPointIndex;
}
```

Task 1

Main

```
#include "my-math.h"
#include <stdio.h>
#include <math.h>

double curveFunction(double x);
double distanceBetweenPoints(double x1, double y1, double x2, double y2);

double EPS = 0.0001;

int main() {

    printf("5! : %.2lf\n", factorial(5));
    printf("2^3 : %.2lf\n", pow(2,3));

    printf("sin(1) : %lf\n", my_sin(1, EPS));
    printf("cos(1) : %lf\n", my_cos(1, EPS));
    printf("f(x) : %lf\n", curveFunction(1));

    int from, to, count;

    printf("Enter from: ");
    scanf("%d", &from);
    printf("Enter to: ");
    scanf("%d", &to);
    printf("Enter count: ");
    scanf("%d", &count);

    double stepLength = (double)(to - from) / count;

    double curveLength = 0;
    double currentPos = from;

    int i;
    for (i = 0; i < count; i++){
        curveLength += distanceBetweenPoints(currentPos,
curveFunction(currentPos), currentPos - stepLength, curveFunction(currentPos +
stepLength));
        currentPos += stepLength;
    }

    printf("Curve length: %lf\n", curveLength);

    return 0;
}

double curveFunction(double x){
    return pow(my_sin(x, EPS),2) / (1 + pow(my_cos(x, EPS),3));
}

double distanceBetweenPoints(double x1, double y1, double x2, double y2){
    return sqrt(pow((x2 - x1),2) + pow((y2 - y1), 2)) ;
```

```
}
```

Task 2

Main

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>

double* generateArray(int length, double min, double max);
void printArray(double * array, int length);
double calculateArithmeticalMean(double* array, int length);
int findCountOfNumbersBiggerThanNumber(double* array, int length, double number);
int findIndexOfFirstNegative(double* array, int length);
double absSumOfArrayElements(double* array, int length, int startIndex);

int main() {
   int arrayLength = 10;
   srand(time(NULL));

   double* array = generateArray(arrayLength, -10, 10);
   printArray(array, arrayLength);

   // Arithmetical mean of array
   double arithmeticalMeanOfArray = calculateArithmeticalMean(array, arrayLength);
   int countOfElementsBiggerThanArithmeticalMean = findCountOfNumbersBiggerThanNumber(array, arrayLength, arithmeticalMeanOfArray);
```

```
        printf("Arithmetical mean is %lf\n", arithmeticalMeanOfArray);
        printf("Count of elements bigger than arithmetical mean is %d\n\n",
countOfElementsBiggerThanArithmeticalMean);

        int indexOfFirstNegativeElement = findIndexOfFirstNegative(array,
arrayLength);

        if (indexOfFirstNegativeElement != -1) {
            double absSumOfElements = absSumOfArrayElements(array,
arrayLength, indexOfFirstNegativeElement);
            printf("Sum of elements after first negative number is %lf\n",
absSumOfElements);
        } else {
            printf("Sum of elements after first negative number is 0\n");
        }

        return 0;
    }

    double* generateArray(int length, double min, double max) {
        double * array = (double*)malloc(sizeof(double) * length);

        int index;

        for (index = 0 ; index < length ; index++) {
            array[index] = (double)rand() * (max - min) / (double)RAND_MAX +
min;
        }

        return array;
```

```
}

void printArray(double* array, int length) {
    int index;
    for (index = 0; index < length; index++) {
        printf("%.2lf ", array[index]);
    }
    printf("\n");
}

double calculateArithmeticalMean(double* array, int length) {
    int index;
    double sumOfNumbers = 0;

    for (index = 0; index < length; index++) {
        sumOfNumbers += array[index];
    }

    return sumOfNumbers / length;
}

int findCountOfNumbersBiggerThanNumber(double* array, int length, double number) {
    int index;
    int countOfElements = 0;

    for (index = 0; index < length; index++) {
        if (array[index] > number) {
            countOfElements++;
        }
    }

    return countOfElements;
}

int findIndexOfFirstNegative(double* array, int length) {
    int index;

    for (index = 0; index < length; index++) {
        if (array[index] < 0) {
            return index;
        }
    }

    return -1;
}

double absSumOfArrayElements(double* array, int length, int startIndex) {
    int index;
    double sumOfElements = 0;

    for (index = startIndex; index < length ; index++) {
        sumOfElements += fabs(array[index]);
    }

    return sumOfElements;
}
```

Task 3

Main

```
#include <stdlib.h>
```

```
#include <ctype.h>
#include <string.h>
#include <stdio.h>
#include <time.h>

#define NUMBER_OF_SUBJECTS 3

typedef struct {
   double mark;
   char subjectName[30];
} Subject;

typedef struct {
   char lastName[12];
   int group;
   Subject subjects[NUMBER_OF_SUBJECTS];
} Student;

Student* generateStudentInfo(int length, int maxNumberOfGroups);
char* generateRandomString(int length);
void printAllStudentsInfo(Student* students, int length);
void printLastNameAllStudentWhoHaveMarkLessThanNumber(Student* students, int length, int number);
int numberOfStudentsWhoHaveOnlyWellOrExcellentMarks(Student* students, int length);
void printBestPassedStudentsSubject(Student* students, int length);
int worstPerformanceGroupNumber(Student* students, int length, int maxNumberOfGroups);
double calculateAverageGroupPerformance(Student* students, int length, int groupNumber);
```

```
    int main() {
        srand(time(NULL));

        int length = 10;
        int maxNumberOfGroups = 4;
        Student* students = generateStudentInfo(length, maxNumberOfGroups);

        printAllStudentsInfo(students, length);

        printf("STUDENTS THAT HAS MARK LESS THEN 50\n");
        printLastNameAllStudentWhoHaveMarkLessThanNumber(students,
length, 50);
        printf("\n\n");

        printf("PERCENTAGE OF STUDENTS THAT HAVE ONLY WELL OR
EXCELLENT MARKS IS %.2lf\n\n",

(double)(numberOfStudentsWhoHaveOnlyWellOrExcellentMarks(students,
length) * 100) / length);

        printf("BEST PASSED SUBJECTS BY STUDENTS");
        printBestPassedStudentsSubject(students, length);
        printf("\n");

        printf("THE WORST PERFORMANCE GROUP IS %d\n",
worstPerformanceGroupNumber(students, length, maxNumberOfGroups));
        return 0;
    }
```

```
Student* generateStudentInfo(int length, int maxNumberOfGroups) {
    Student* students = (Student*) malloc(sizeof(Student) * length);

    int index;
    for (index = 0; index < length; index++) {
        // set student last name
        char* lastName = generateRandomString(rand() % 6 + 7);
        lastName[0] = toupper(lastName[0]);
        strcpy(students[index].lastName, lastName);

        // set student group
        students[index].group = rand() % maxNumberOfGroups + 1;

        // set student subject marks
        students[index].subjects[0].mark = rand() % 100 + 1;
        strcpy(students[index].subjects[0].subjectName, "Web Technologies");
        students[index].subjects[1].mark = rand() % 100 + 1;
        strcpy(students[index].subjects[1].subjectName, "Ukrainian language");
        students[index].subjects[2].mark = rand() % 100 + 1;
        strcpy(students[index].subjects[2].subjectName, "Information System
Design");
    }

    return students;
}

// according to
// https://stackoverflow.com/questions/15767691/whats-the-c-library-
function-to-generate-random-string
char* generateRandomString(int length) {
    // available chars for generating random last name
    char charset[] = "abcdefghijklmnopqrstuvwxyz";
    char* randomString = (char*) malloc(sizeof(char) * length);

    int index;
    for (index = 0; index < length - 1; index++) {
        int randomCharsetIndex = rand() % (sizeof(charset) - 1);
        randomString[index] = charset[randomCharsetIndex];
    }

    randomString[length - 1] = '\0';
    return randomString;
}

void printAllStudentsInfo(Student* students, int length) {
    int i, j;
    for (i = 0; i < length; i++) {
        printf("Last name: %s\n", students[i].lastName);
        printf("Group: %d\n", students[i].group);
        printf("Subjects\n");
        for (j = 0; j < NUMBER_OF_SUBJECTS; j++) {
            printf("|Subject:            %-30s|Mark:            %5.2lf|\n",
students[i].subjects[j].subjectName, students[i].subjects[j].mark);
        }
        printf("\n");
    }
}

void    printLastNameAllStudentWhoHaveMarkLessThanNumber(Student*
students, int length, int number) {
    int i, j;
    for (i = 0; i < length; i++) {
        for (j = 0; j < NUMBER_OF_SUBJECTS; j++) {
            if (students[i].subjects[j].mark < number) {
                printf("%s\n", students[i].lastName);
                break;
            }
        }
    }
}

int numberOfStudentsWhoHaveOnlyWellOrExcellentMarks(Student* students, int length) {
    int counter = 0;
    int i, j;
    for (i = 0; i < length; i++) {
        int allMarksWellOrExcellent = 1;
        for (j = 0; j < NUMBER_OF_SUBJECTS; j++) {
            if (students[i].subjects[j].mark >= 50) {
                allMarksWellOrExcellent = 0;
                break;
            }
        }

        if (allMarksWellOrExcellent) {
            counter++;
        }
    }

    return counter;
```

```
}

void printBestPassedStudentsSubject(Student* students, int length) {
    int i, j;
    for (i = 0; i < length; i++) {
        int indexOfBestPassedSubject = 0;
        for (j = 1; j < NUMBER_OF_SUBJECTS; j++) {
            if (students[i].subjects[j].mark > students[i].subjects[indexOfBestPassedSubject].mark) {
                indexOfBestPassedSubject = j;
            }
        }

        printf("Best passed subject by student '%s' is %s\n", students[i].lastName, students[i].subjects[indexOfBestPassedSubject].subjectName);
    }
}

int worstPerformanceGroupNumber(Student* students, int length, int maxNumberOfGroups) {
    int groupNumber;
    int worstGroupNumber = 1;
    int worstGroupAverageMarks = calculateAverageGroupPerformance(students, length, 1);

    for (groupNumber = 2; groupNumber <= maxNumberOfGroups; groupNumber++) {
        int groupAverageMarks = calculateAverageGroupPerformance(students, length, groupNumber);
        if (groupAverageMarks < worstGroupAverageMarks) {
            worstGroupAverageMarks = groupAverageMarks;
            worstGroupNumber = groupNumber;
        }
    }

    return worstGroupNumber;
}

double calculateAverageGroupPerformance(Student* students, int length, int
groupNumber) {
    int i, j;
    int countOfSubjects = 0;
    double sumOfMarks = 0;

    for (i = 0; i < length; i++) {
        if (students[i].group == groupNumber) {
            for (j = 0; j < NUMBER_OF_SUBJECTS; j++) {
                countOfSubjects++;
                sumOfMarks += students[i].subjects[j].mark;
            }
        }
    }

    return sumOfMarks / countOfSubjects;
}
```

Task 4

Main

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

```
#include <time.h>
#include "array-list.h"

void fillList(ArrayList* list, int length);
void printList(ArrayList *list);

int main() {

    srand(time(NULL));

    ArrayList* list = newArrayList();

    printf("Using function ADD (10 times)\n");
    fillList(list, 10);
    printList(list);
    printf("\n");

    printf("Using function INSERT (4 times)\n");
    insert(list, 5, 101);
    insert(list, 6, 102);
    insert(list, 7, 103);
    insert(list, 8, 104);
    printList(list);
    printf("\n");

    printf("Using function REMOVE (2 times)\n");
    removeItem(list, list->length - 1);
    removeItem(list, list->length - 1);
    printList(list);
    printf("\n");
    printf("Using function SET (2 times)\n");
    set(list, 0, 1000);
    set(list, 1, 1001);
    printList(list);
    printf("\n");

    printf("Using function GET (2 times)\n");
    printf("Element by index 2 is %d\n", get(list, 2));
    printf("Element by index 5 is %d\n", get(list, 5));
    printf("\n");

    printf("Using function SIZE\n");
    printf("Size is %d\n\n", size(list));

    return 0;
}

void fillList(ArrayList* list, int length) {
    int index;
    for (index = 0; index < length; index++) {
        add(list, rand() % 10);
    }
}

void printList(ArrayList* list) {
    int index;
    for (index = 0; index < list->length; index++) {
        printf("%d ", get(list, index));
    }
    printf("\n");
}
```

Task 5

Main

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include "linked-list.h"

void fillList(LinkedList* list, int length);
void printList(LinkedList* list);

int main() {
    srand(time(NULL));

    LinkedList* list = newLinkedList();

    printf("Using function ADD (10 times)\n");
    fillList(list, 10);
    printList(list);
    printf("\n");

    printf("Using function INSERT (4 times)\n");
    insert(list, 5, 101);
    insert(list, 6, 102);
    insert(list, 7, 103);
    insert(list, 8, 104);
    printList(list);
    printf("\n");
    printf("Using function REMOVE (2 times)\n");
    removeNode(list, list->length - 1);
    removeNode(list, list->length - 1);
    printList(list);
    printf("\n");

    printf("Using function SET (2 times)\n");
    set(list, 0, 1000);
    set(list, 1, 1001);
    printList(list);
    printf("\n");

    printf("Using function GET (2 times)\n");
    printf("Element by index 2 is %d\n", get(list, 2));
    printf("Element by index 4 is %d\n", get(list, 4));
    printf("\n");

    printf("Using function SIZE\n");
    printf("Size is %d\n\n", size(list));

    return 0;
}

void fillList(LinkedList* list, int length) {
    int index;
    for (index = 0; index < length; index++) {
        add(list, rand() % 10);
    }
}
```

```
void printList(LinkedList* list) {
    int index;
    for (index = 0; index < list->length; index++) {
        printf("%d ", get(list, index));
    }
    printf("\n");
}
```

Task 6

Main

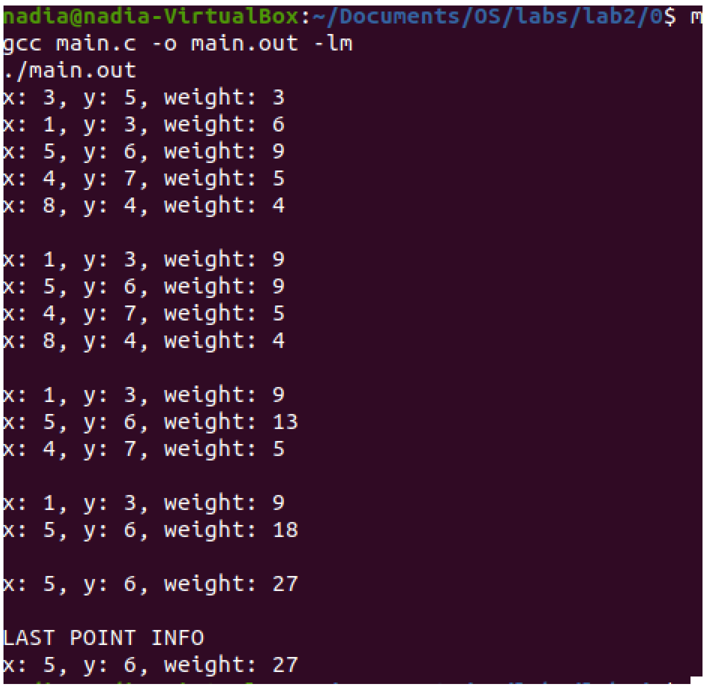
```
#include <stdio.h>

#include "matrix.h"

int main() {

    printf("FILL FIRST MATRIX:\n");
    Matrix* matrix1 = newMatrix(2, 2);
    fillMatrixThroughTerminal(matrix1);
    printf("\n");

    printf("SAVE MATRIX TO FILE...\n");
    saveMatrixToFile(matrix1,"matrix.txt");

    printf("LOAD MATRIX FROM FILE...\n");
    Matrix* savedMatrix = loadMatrixFromFile("matrix.txt");

    printf("LOADED MATRIX:\n");
    printMatrix(savedMatrix);
    printf("\n");
```

```
	printf("FILL SECOND MATRIX:\n");
	Matrix* matrix2 = newMatrix(2, 2);
	fillMatrixThroughTerminal(matrix2);
	printf("\n");

	printf("FIRST MATRIX:\n");
	printMatrix(matrix1);
	printf("SECOND MATRIX:\n");
	printMatrix(matrix2);
	printf("\n");

	printf("SUM OF TWO MATRIX:\n");
	Matrix* sumMatrix = sum(matrix1, matrix2);
	printMatrix(sumMatrix);
	printf("\n");

	printf("SUBTRACTION OF MATRIX:\n");
	Matrix* subtractionMatrix = subtraction(matrix1, matrix2);
	printMatrix(subtractionMatrix);
	printf("\n");

	printf("USING SET MATRIX METHOD TO INDEX [0, 0] (1 time)\n");
	set(matrix1,0,0,1337);
	printf("ELEMENT WITH INDEXES [0, 0] is %.2lf\n\n", get(matrix1, 0,
0));

	printf("USING RESIZE MATRIX METHOD (2 times for 2 matrix)\n");
	resizeMatrix(matrix1,3,2);
	resizeMatrix(matrix2,2,3);
	printf("FIRST MATRIX\n");
	printMatrix(matrix1);
	printf("SECOND MATRIX\n");
	printMatrix(matrix2);
	printf("\n");

	printf("USING MULTIPLICATION MATRIX METHOD\n");
	Matrix* multiplicationMatrix = multiplication(matrix1,matrix2);

	printf("MULTIPLICATED MATRIX:\n");
	printMatrix(multiplicationMatrix);
	printf("\n");

	printf("USING MULTIPLICATION MATRIX BY NUMBER
METHOD:\n");
	printf("MULTIPLICATED BY NUMBER MATRIX:\n");
	Matrix* multiplicationMatrixByNumber =
multiplicationByNumber(matrix1,5);
	printMatrix(multiplicationMatrixByNumber);
	printf("\n");

	printf("USING CLEAR MATRIX (6 times)\n");
	clearMatrix(matrix1);
	clearMatrix(matrix2);
	clearMatrix(sumMatrix);
	clearMatrix(subtractionMatrix);
	clearMatrix(multiplicationMatrix);
	clearMatrix(multiplicationMatrixByNumber);

	return 0;
```

}

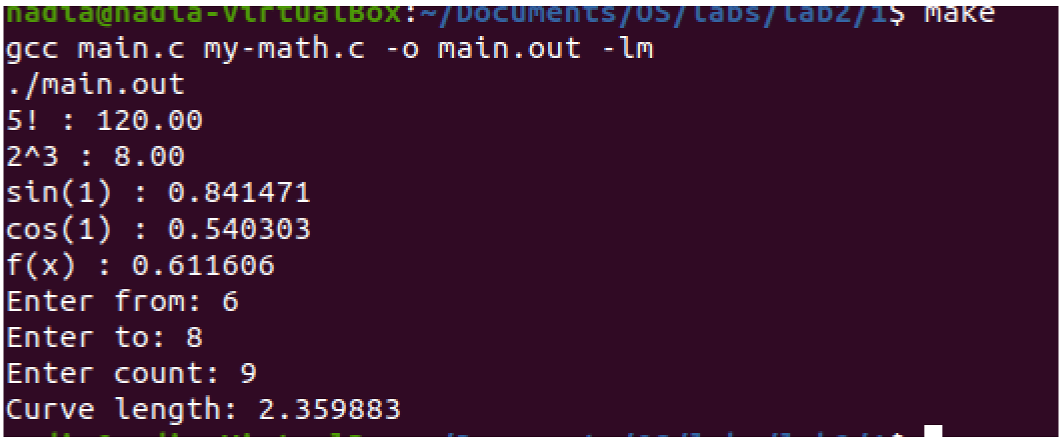
Результат:

Task 0



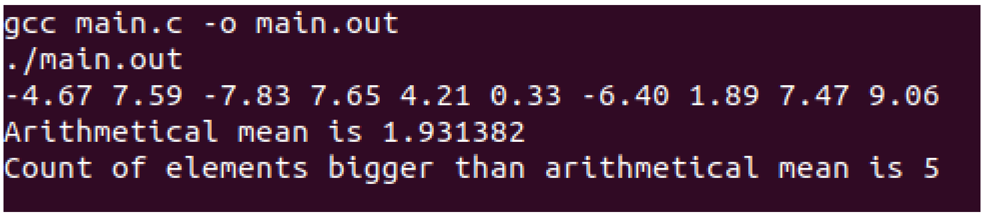
```
nadia@nadia-VirtualBox:~/Documents/OS/labs/lab2/0$ m
gcc main.c -o main.out -lm
./main.out
x: 3, y: 5, weight: 3
x: 1, y: 3, weight: 6
x: 5, y: 6, weight: 9
x: 4, y: 7, weight: 5
x: 8, y: 4, weight: 4

x: 1, y: 3, weight: 9
x: 5, y: 6, weight: 9
x: 4, y: 7, weight: 5
x: 8, y: 4, weight: 4

x: 1, y: 3, weight: 9
x: 5, y: 6, weight: 13
x: 4, y: 7, weight: 5

x: 1, y: 3, weight: 9
x: 5, y: 6, weight: 18

x: 5, y: 6, weight: 27

LAST POINT INFO
x: 5, y: 6, weight: 27
```

Task 1

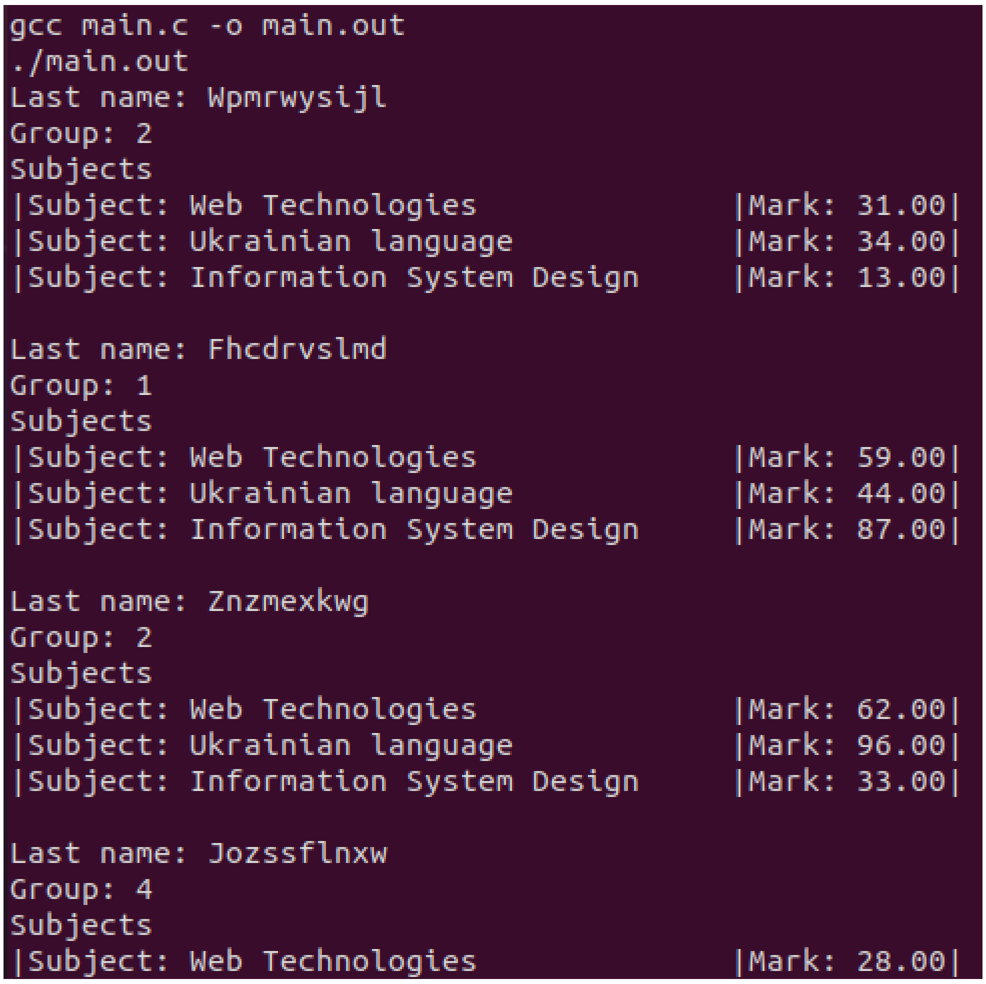


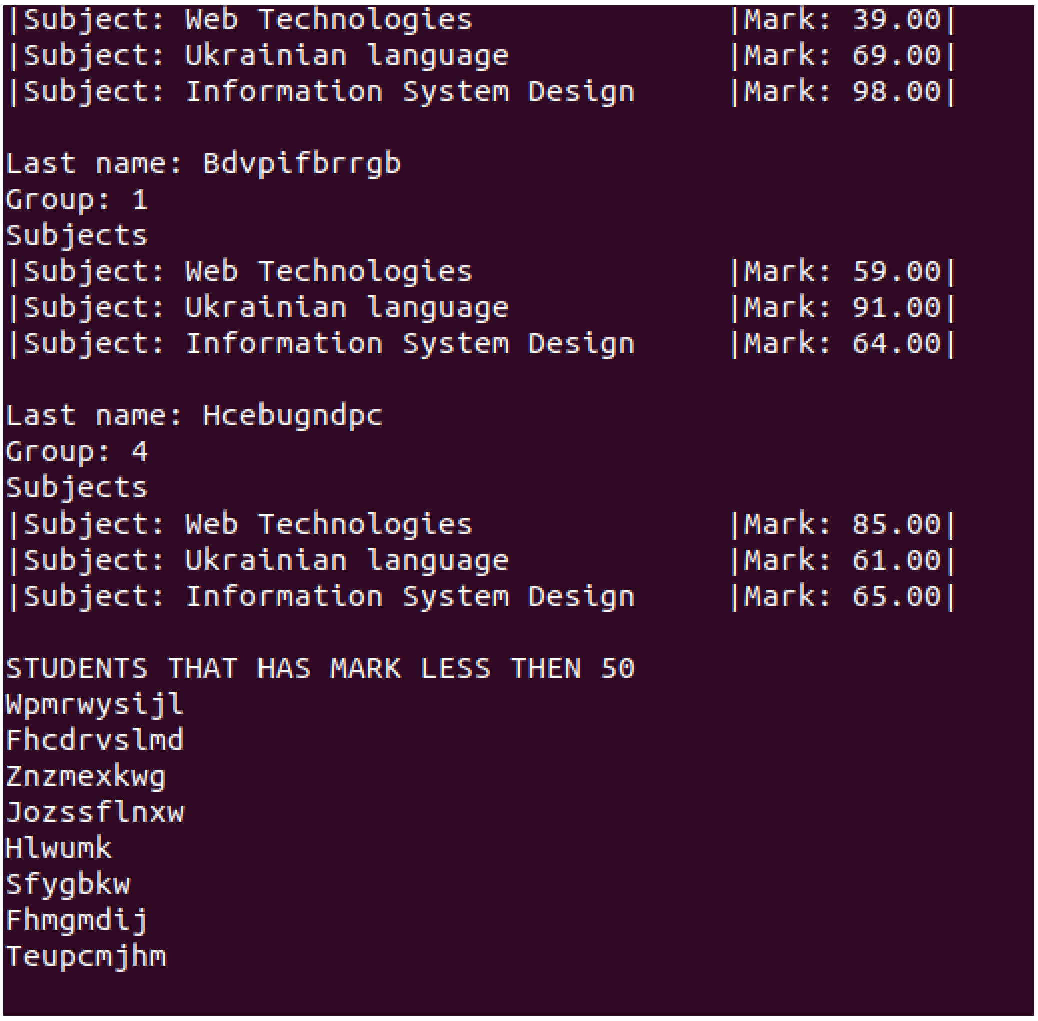
```
nadia@nadia-VirtualBox:~/Documents/OS/labs/lab2/1$ make
gcc main.c my-math.c -o main.out -lm
./main.out
5! : 120.00
2^3 : 8.00
sin(1) : 0.841471
cos(1) : 0.540303
f(x) : 0.611606
Enter from: 6
Enter to: 8
Enter count: 9
Curve length: 2.359883
```

Task 2

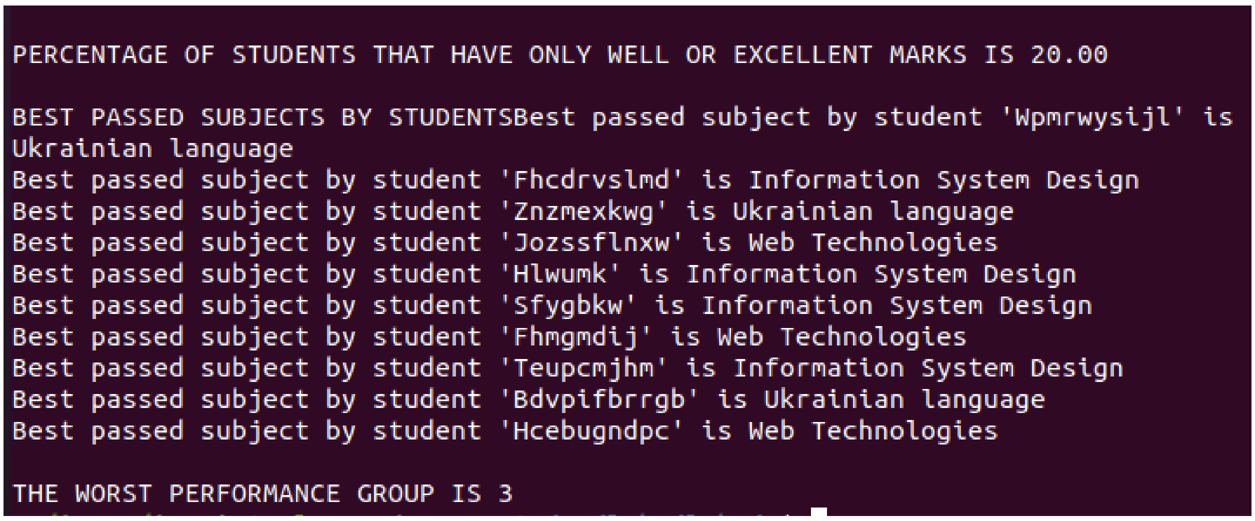

```
gcc main.c -o main.out
./main.out
-4.67 7.59 -7.83 7.65 4.21 0.33 -6.40 1.89 7.47 9.06
Arithmetical mean is 1.931382
Count of elements bigger than arithmetical mean is 5
```

Task 3

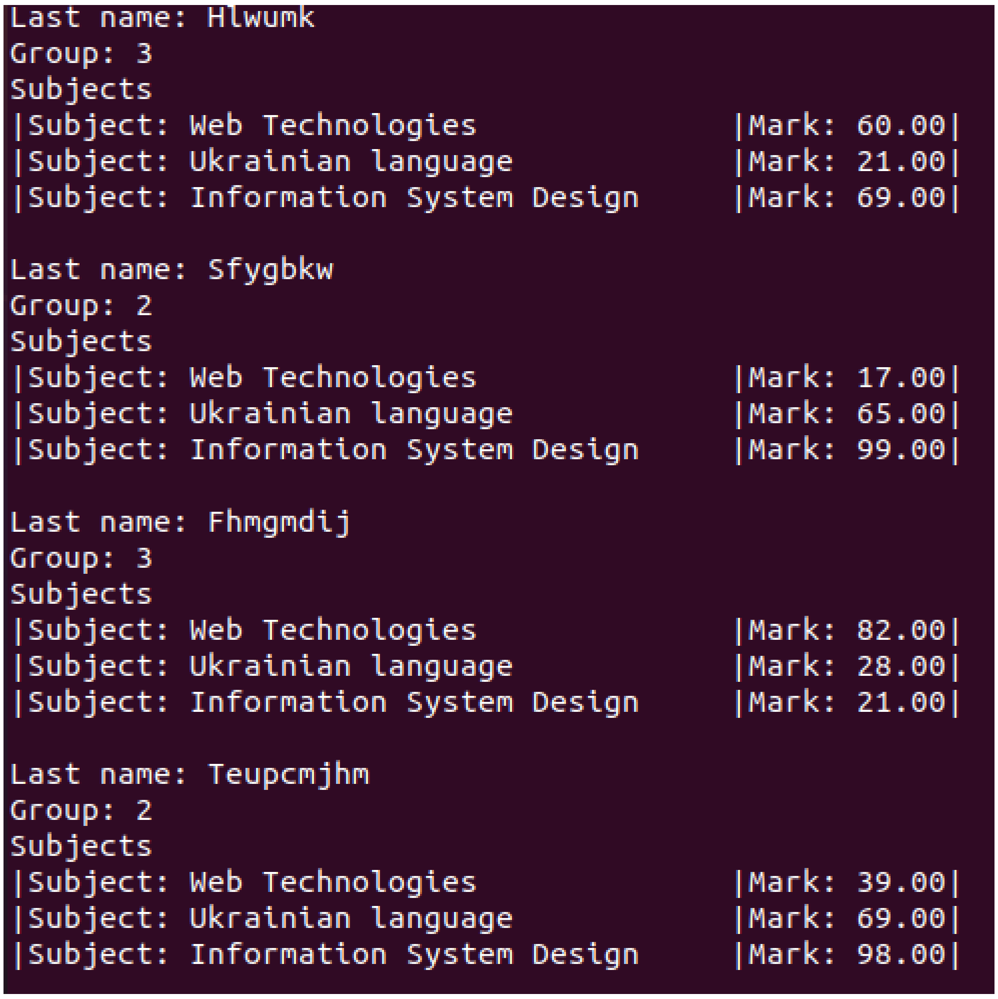

```
gcc main.c -o main.out
./main.out
Last name: Wpmrwysijl
Group: 2
Subjects
|Subject: Web Technologies                    |Mark: 31.00|
|Subject: Ukrainian language                  |Mark: 34.00|
|Subject: Information System Design           |Mark: 13.00|

Last name: Fhcdrvslmd
Group: 1
Subjects
|Subject: Web Technologies                    |Mark: 59.00|
|Subject: Ukrainian language                  |Mark: 44.00|
|Subject: Information System Design           |Mark: 87.00|

Last name: Znzmexkwg
Group: 2
Subjects
|Subject: Web Technologies                    |Mark: 62.00|
|Subject: Ukrainian language                  |Mark: 96.00|
|Subject: Information System Design           |Mark: 33.00|

Last name: Jozssflnxw
Group: 4
Subjects
|Subject: Web Technologies                    |Mark: 28.00|
```



```
|Subject: Web Technologies              |Mark: 39.00|
|Subject: Ukrainian language            |Mark: 69.00|
|Subject: Information System Design     |Mark: 98.00|

Last name: Bdvpifbrrgb
Group: 1
Subjects
|Subject: Web Technologies              |Mark: 59.00|
|Subject: Ukrainian language            |Mark: 91.00|
|Subject: Information System Design     |Mark: 64.00|

Last name: Hcebugndpc
Group: 4
Subjects
|Subject: Web Technologies              |Mark: 85.00|
|Subject: Ukrainian language            |Mark: 61.00|
|Subject: Information System Design     |Mark: 65.00|

STUDENTS THAT HAS MARK LESS THEN 50
Wpmrwysijl
Fhcdrvslmd
Znzmexkwg
Jozssflnxw
Hlwumk
Sfygbkw
Fhmgmdij
Teupcmjhm
```

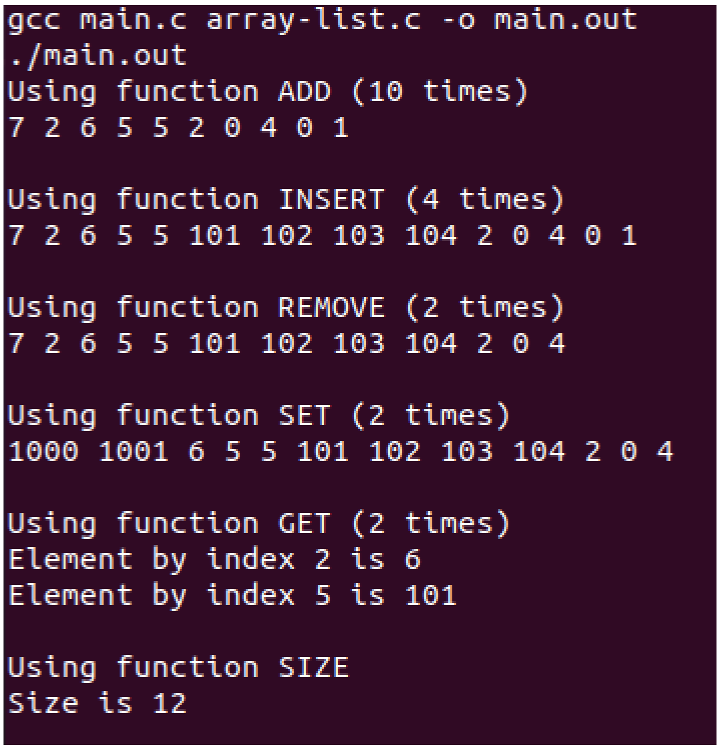


```
PERCENTAGE OF STUDENTS THAT HAVE ONLY WELL OR EXCELLENT MARKS IS 20.00

BEST PASSED SUBJECTS BY STUDENTSBest passed subject by student 'Wpmrwysijl' is
Ukrainian language
Best passed subject by student 'Fhcdrvslmd' is Information System Design
Best passed subject by student 'Znzmexkwg' is Ukrainian language
Best passed subject by student 'Jozssflnxw' is Web Technologies
Best passed subject by student 'Hlwumk' is Information System Design
Best passed subject by student 'Sfygbkw' is Information System Design
Best passed subject by student 'Fhmgmdij' is Web Technologies
Best passed subject by student 'Teupcmjhm' is Information System Design
Best passed subject by student 'Bdvpifbrrgb' is Ukrainian language
Best passed subject by student 'Hcebugndpc' is Web Technologies

THE WORST PERFORMANCE GROUP IS 3
```



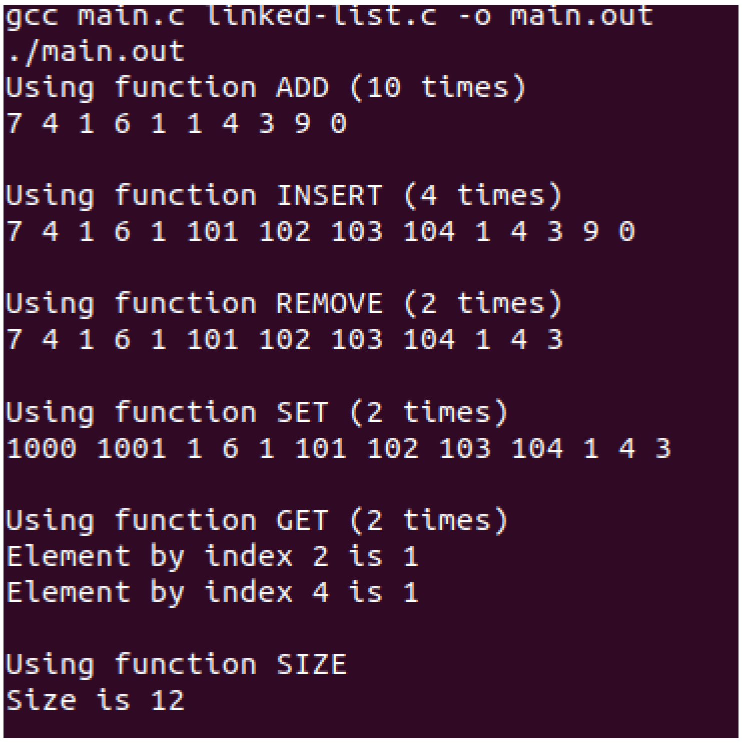
```
Last name: Hlwumk
Group: 3
Subjects
|Subject: Web Technologies              |Mark: 60.00|
|Subject: Ukrainian language            |Mark: 21.00|
|Subject: Information System Design     |Mark: 69.00|

Last name: Sfygbkw
Group: 2
Subjects
|Subject: Web Technologies              |Mark: 17.00|
|Subject: Ukrainian language            |Mark: 65.00|
|Subject: Information System Design     |Mark: 99.00|

Last name: Fhmgmdij
Group: 3
Subjects
|Subject: Web Technologies              |Mark: 82.00|
|Subject: Ukrainian language            |Mark: 28.00|
|Subject: Information System Design     |Mark: 21.00|

Last name: Teupcmjhm
Group: 2
Subjects
|Subject: Web Technologies              |Mark: 39.00|
|Subject: Ukrainian language            |Mark: 69.00|
|Subject: Information System Design     |Mark: 98.00|
```

Task 4



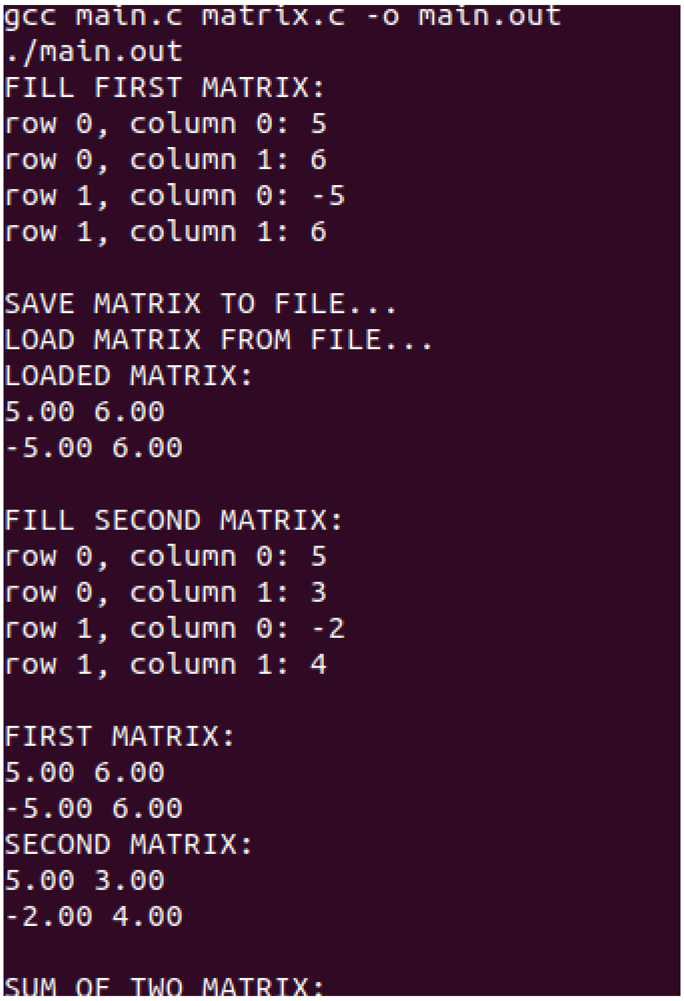
```
gcc main.c array-list.c -o main.out
./main.out
Using function ADD (10 times)
7 2 6 5 5 2 0 4 0 1

Using function INSERT (4 times)
7 2 6 5 5 101 102 103 104 2 0 4 0 1

Using function REMOVE (2 times)
7 2 6 5 5 101 102 103 104 2 0 4

Using function SET (2 times)
1000 1001 6 5 5 101 102 103 104 2 0 4

Using function GET (2 times)
Element by index 2 is 6
Element by index 5 is 101

Using function SIZE
Size is 12
```

Task 5

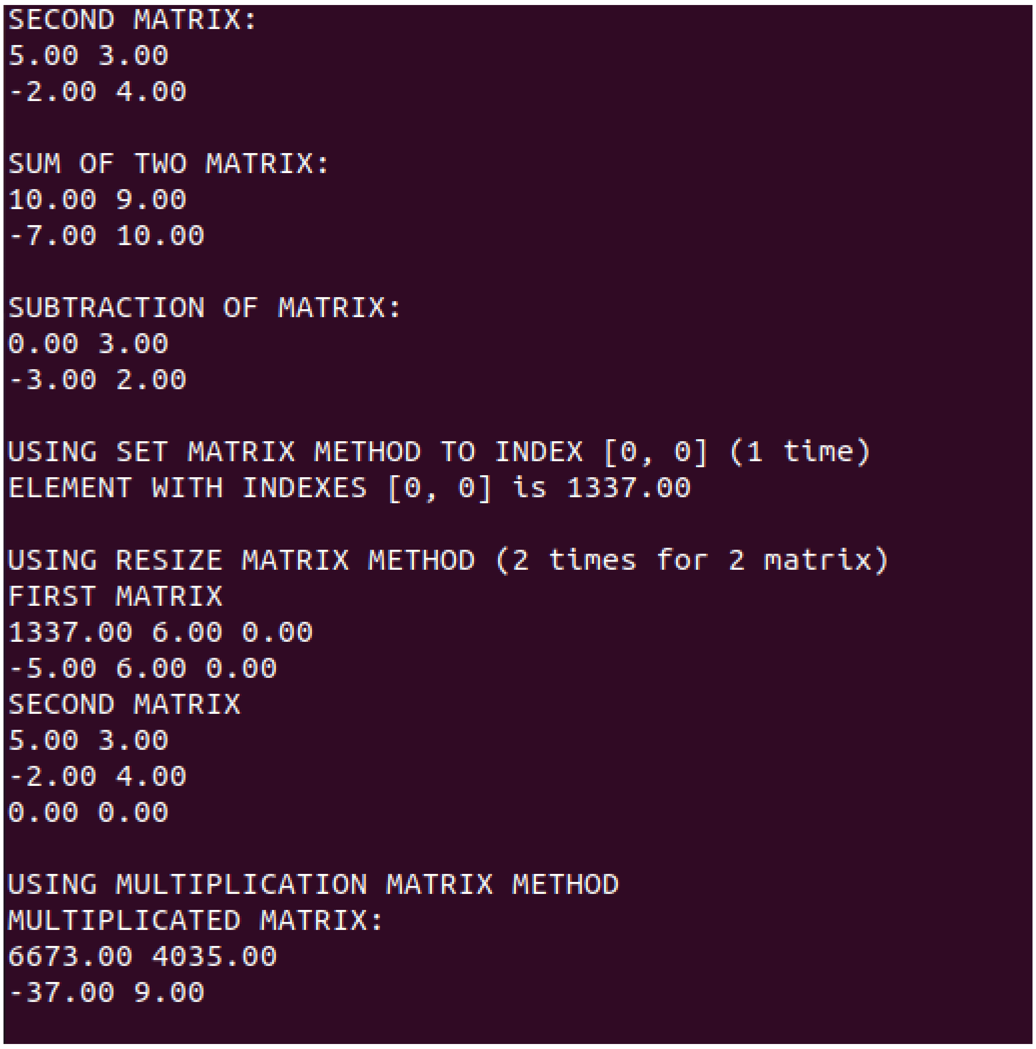


```
gcc main.c linked-list.c -o main.out
./main.out
Using function ADD (10 times)
7 4 1 6 1 1 4 3 9 0

Using function INSERT (4 times)
7 4 1 6 1 101 102 103 104 1 4 3 9 0

Using function REMOVE (2 times)
7 4 1 6 1 101 102 103 104 1 4 3

Using function SET (2 times)
1000 1001 1 6 1 101 102 103 104 1 4 3

Using function GET (2 times)
Element by index 2 is 1
Element by index 4 is 1

Using function SIZE
Size is 12
```

Task 6



```
gcc main.c matrix.c -o main.out
./main.out
FILL FIRST MATRIX:
row 0, column 0: 5
row 0, column 1: 6
row 1, column 0: -5
row 1, column 1: 6

SAVE MATRIX TO FILE...
LOAD MATRIX FROM FILE...
LOADED MATRIX:
5.00 6.00
-5.00 6.00

FILL SECOND MATRIX:
row 0, column 0: 5
row 0, column 1: 3
row 1, column 0: -2
row 1, column 1: 4

FIRST MATRIX:
5.00 6.00
-5.00 6.00
SECOND MATRIX:
5.00 3.00
-2.00 4.00

SUM OF TWO MATRIX:
```


```
SECOND MATRIX:
5.00 3.00
-2.00 4.00

SUM OF TWO MATRIX:
10.00 9.00
-7.00 10.00

SUBTRACTION OF MATRIX:
0.00 3.00
-3.00 2.00

USING SET MATRIX METHOD TO INDEX [0, 0] (1 time)
ELEMENT WITH INDEXES [0, 0] is 1337.00

USING RESIZE MATRIX METHOD (2 times for 2 matrix)
FIRST MATRIX
1337.00 6.00 0.00
-5.00 6.00 0.00
SECOND MATRIX
5.00 3.00
-2.00 4.00
0.00 0.00

USING MULTIPLICATION MATRIX METHOD
MULTIPLICATED MATRIX:
6673.00 4035.00
-37.00 9.00
```