**Комплексные числа. Реализовать модуль для работы с комплексными числами.**

Модуль должен содержать следующие функции и структуры:

// Комплексное число

typedef struct

{

float x;

float y;

} Complex;

// Сложение

Complex add(Complex a, Complex b);

// Умножение

Complex mul(Complex a, Complex b);

// Деление

Complex div(Complex a, Complex b);

// Получение модуля

float mod(Complex a);

// Получение аргумента

float add(Complex a);

// Получение сопряжённого

Complex add(Complex a);

// Вывод в алгебраической форме

void printAlg(Complex a);

// Вывод в тригонометрической форме

void printTrig(Complex a);

// Вывод в показательной форме

void printExp(Complex a)

**В файле записана информация о зачете в текстовом виде (фамилии латиницей в транслите, unix перевод строки). Информация о студенте описывается структурой:**

typedef struct

{

char lastName[64]; // фамилия

int id; // шифр

int mark; // оценка: 0 - fail, 1 - pass

} Student;

− Название файла передается первым параметром командной строки.

− В файле первой строкой идет количество студентов.

− В последующих:

− фамилия, записанная наоборот

− шифр

− оценка: `0` или `1`

Например:

vogedanzeB

133742

0

1) Сделать функцию разворачивания строки void reverse(char\* string).

2) Сделать функцию вывода void print(const Student\* student), которая выводит информацию об одном студенте в формате:

LastName: фамилия (в правильном виде)

Id: шифр

Mark: оценка (в виде слова. Если 0, то fail. Если 1, то pass)

3) Вывести информацию о студентах в stdout. Фамилия должна быть в правильном (развернутом) виде. Оценка в виде слова (`fail`, `pass`).

4) В конце распечатать процент сдавших в формате:

Passed: процент сдавших

5) При успешном выполнении программы в качестве кода возврата возвращать `0`. В случае отсутствия аргументов командной строки, отсутствия файла, ошибки чтения файла (количество данных меньше чем указано в первой строке файла) в качестве кода возврата возвращать `1`.

Пример ввода:

2

vonavI

156256

1

vogedanzeB

133742

0

Пример вывода:

LastName: Ivanov

Id: 156256

Mark: pass

LastName: Beznadegov

Id: 133742

Mark: fail

Passed: 50.000000

Программа позволять создавать, редактировать и просматривать контакты. Количество контактов заранее неизвестно. Каждый контакт содержит:  
1. «Имя» – до 80 символов,  
2. «Номер телефона» – до 12 символов,  
3. «Группа» – одно из: нет группы, семья, друзья, коллеги.  
  
Программа выводит на консоль меню с вариантами действий, ожидает, пока пользователь введет символ, обозначающий какую либо команду, выполнят ее, и, если команда не означает завершение работы, опять выводит меню и тд. Варианты действий:  
  
«Добавить контакт» – запрашиваем данные для полей и сохраняем, имена контактов должны быть уникальны  
«Удалить контакт» – запрашиваем имя и удаляем контакт с таким именем  
«Изменить имя/телефон/группу» – делаем подменю, спрашиваем имя, потом что делать и новое значение  
«Вывести» – делаем подменю, спрашиваем как выводить, и, если надо, что искать:  
контакт по имени – запрашиваем имя и красиво выводим в консоль данные,  
контакт по телефону,  
все контакты,  
все контакты из указанной группы.  
«Завершить работу».  
Имя группы пользователь вводит строкой, в структуре хранить с помощью enum. Пример:  
  
enum Groups  
{  
NO\_GROUP,  
FAMILY,  
FRIENDS,  
COLLEAGUES  
};  
// структура контакта:  
struct Contact  
{  
char name[80];  
char phone[12];  
int group;  
}  
// пример работы с enum  
Contact c;  
[c.group](https://vk.com/away.php?to=http%3A%2F%2Fc.group&cc_key=) = FRIENDS;  
​  
switch (group)  
{  
case NO\_GROUP:  
printf("No group");  
​  
case FRIENDS:  
printf("Frends");  
​  
...  
}

Реализовать функции по работе со строками

//делает строку числом  
int strToInt(const char\* str);

//Перевод в нижний регистр, изменяя строку  
void mutableToLower (char\* str);

//Перевод в нижний регистр, не изменяя строку  
char\* immutableToLower (const char\* str);

//Перевод в верхний регистр, изменяя строку   
void mutableToUpper (char\* str);

//Перевод в верхний регистр, не изменяя строку   
char\* immutableToUpper (const char\* str);

//удаляет пробелы в начале и конце строки, изменяя ее  
void mutableStrip (char\* str);

//удаляет пробелы в начале и конце строки, не изменяя ее   
char\* immutableStrip (const char\* str);

// удаляет все пробелы в строке, изменяя ее  
void mutableRemoveSpaces (char\* str);

// удаляет все пробелы в строке, не изменяя ее   
char\* immutableRemoveSpaces (const char\* str);

// Удаляет из строки все, кроме чисел и букв, изменяя ее  
void mutableFilter (char\* str);

// Удаляет из строки все, кроме чисел и букв, не изменяя ее   
char\* immutableFilter (const char\* str);

// Проверяет, является ли строка числом  
bool isNumber (const char\* str);

// Проверяет, является ли строка набором букв  
bool isAlpha (const char\* str);