Написать функцию, которая читает массив из файла, сортирует его указанным методом, записывает отсортированный массив в новый файл.

Вариант	Функция
1	1. Сортировка пузырьком. 2. Быстрая сортировка.
2	1. Сортировка вставками. 2. Heap sort.
3	1. Сортировка выбором. 2. Сортировка слиянием.
4	1. Шейкерная сортировка. 2. Поразрядная сортировка.
5	1. Сортировка Шелла (с последовательностью шагов Седжвика 1, 5, …). 2. Поразрядная сортировка.
6	1. Сортировка подсчётом. 2. Быстрая сортировка.
7	1. Сортировка пузырьком. 2. Heap sort.
8	1. Сортировка вставками. 2. Сортировка слиянием.
9	1. Сортировка выбором. 2. Быстрая сортировка.
10	1. Шейкерная сортировка. 2. Heap sort.
11	1. Сортировка Шелла (с последовательностью шагов Седжвика 1, 5, …). 2. Сортировка слиянием.
12	1. Сортировка подсчётом. 2. Поразрядная сортировка.
13	1. Сортировка пузырьком. 2. Сортировка слиянием.
14	1. Сортировка вставками. 2. Поразрядная сортировка.
15	1. Сортировка выбором. 2. Heap sort.
16	1. Шейкерная сортировка. 2. Быстрая сортировка.
17	1. Сортировка Шелла (с последовательностью шагов Ciura 1, 4, 10…). 2. Heap sort.

18	1. Сортировка подсчётом. 2. Сортировка слиянием.
19	1. Сортировка пузырьком. 2. Поразрядная сортировка.
20	1. Сортировка вставками. 2. Быстрая сортировка.
21	1. Сортировка выбором 2. Сортировка слиянием.
22	1. Шейкерная сортировка. 2. Heap sort.
23	1. Сортировка Шелла (с последовательностью шагов Токуда 1, 4, 9, …). 2. Быстрая сортировка.
24	1. Сортировка подсчётом. 2. Heap sort.
25	1. Сортировка расческой. 2. J-сортировка.
26	1. Гномья сортировка. 2. Быстрая сортировка.
27	1. Шейкерная сортировка. 2. J-сортировка.
28	1. Сортировка пузырьком. 2. J-сортировка.
29	1. Сортировка вставками. 2. J-сортировка.
30	1. Сортировка расческой. 2. Поразрядная сортировка.
31	1. Гномья сортировка. 2. J-сортировка.
32	1. Сортировка расческой. 2. Быстрая сортировка.
33	1. Гномья сортировка. 2. Сортировка слиянием.
34	1. Сортировка расческой. 2. Сортировка слиянием.
35	1. Сортировка выбором. 2. J-сортировка.
36	1. Гномья сортировка. 2. Heap sort.

Пример кода:

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

 /*
* Считывание массива целых чисел из файла
*/
int* read_file(char *address, size_t* len){
	FILE* f_in;
	int* arr=0;
	f_in = fopen(address, "r"); //открыть файл, режим "r" - чтение
	if(f_in!=0) // или if(f_in) - файл удалось открыть
	{
		//fgets(str, 80, f_in); - прочесть строку из файла, не более 80 символов или до новой строки, поместить прочитанное в переменную str

		//если нужно прочесть не строку, можно использовать аналог функции scanf(); не забываем указывать файл, откуда читаем

		//считаем, что в начале файла указано количество элементов массива
		fscanf(f_in, "%d", len);

		//выделяем память и читаем массив
		size_t i=0;
		arr = (int*)calloc(*len, sizeof(int));
		while(i<*len)
		{
			fscanf(f_in, "%d", &arr[i]);
			i++;
		}

		//закрыли файл
		fclose(f_in);
	}

	//вернули прочитанный (если всё в порядке) массив
	return arr;
}

 /**
```

```
* Записываем массива целых чисел из файла
**/
int write_to_file(char* address, int* arr, size_t len){
	FILE* f_out;
	f_out = fopen(address, "w"); //открыть файл, режим "w" - запись, "a" - добавить в конец и т.д.

	if(f_out!=0) // или if(f_out) - файл удалось открыть
	{
		//fputs(str, f_out); - записать строку в файл
		//если нужно записать не строку, можно использовать аналог функции printf()

		// записываем в начало файла количество элементов
		fprintf(f_out, "%d", len);

		// записываем массив в файл
		size_t i=0;
		while(i<len)
		{
			fprintf(f_out, " %d", arr[i]);
			i++;
		}

		//закрыли файл
		fclose(f_out);

		//всё хорошо - вернули 1
		return 1;
	}

	//если мы здесь - записать в файл не удалось
	return 0;
}

 /**
* Копируем массив целых чисел
**/
int* copy_array(int* a, size_t len){
	int* arr = (int*)calloc(len, sizeof(int));
	size_t i=0;
	for(;i<len;i++)
	{
		arr[i] = a[i];
	}
	return arr;
}

 /**
* Алгоритм глупой сортировки Stupid sort
```

```
**/
int* stupidsort(int* a, size_t len){
        // копируем массив - будем сортировать его копию
        int* arr = copy_array(a, len);
        size_t i=1;
        int temp;
        for(;i<len;i++)
        {
                if(arr[i-1]>arr[i]){
                        // меняем элементы местами
                        temp = arr[i-1];
                        arr[i-1] = arr[i];
                        arr[i] = temp;
                        // начинаем идти сначала
                        i = 0;
                }
        }
        return arr;
}


int main(){
        // Пути к файлам
        char file_in_path[] = "in.txt"; // "C:\\Users\\Admin\\Documents\\in.txt";
        char file_out_path[] = "out.txt";

        int* arr;
        size_t len;
        size_t i=0;
        char c;

        //читаем массив из входного файла
        arr = read_file(file_in_path, &len);
        if(arr==0)
        {
                printf("\nCouldn't read file. Closing...");
                return 0;
        }
        printf("\nArray from file: \n");
        for(i=0;i<len;i++) printf("%d ", arr[i]);
        printf("\n");

        //сортируем массив
        int* arr_sorted = stupidsort(arr, len);
        printf("\nSorted array: \n");
        for(i=0;i<len;i++) printf("%d ", arr_sorted[i]);
        printf("\n");

        //пишем результат в выходной файл
        printf("\nWrite attempt: %d", write_to_file(file_out_path, arr_sorted, len));
```

```
	printf("\n");

	// не забываем освобождать память
	if(arr!=0)
		free(arr);

	if(arr_sorted!=0)
		free(arr_sorted);

	system("pause");

	return 0;}
```

Файл in.txt:

4 2 0 3 1

4 - количество элементов

2 0 3 1 - сами элементы