Сортировки и алгоритми върху масиви

Insertion sort

1. Напишете функция **bool swapAtIndices(int arrayOfNumbers[], unsigned int i,unsigned int j)**, която разменя стойностите на елементите с индекси **i** и **j**. Пробвайте дали работи правилно. Какво става ако някои от индексите е извън максималния индекс за този масив?

2. Напишете функция void **readAndSort(int resultArray[], unsigned int size),** която приема празен масив **resultArray** и **size –** брой елементи, които трябва да прочете от конзолата. След като завърши функцията, масивът трябва да е запълнен със **size** числа, които са сортирани във възходяш ред. Сортирането да се случва още по време на четенето от конзолата. Използвайте **Insertion sort**.

Bubble sort

3. Имплементирайте **Bubble sort,** използвайте интернет и асистентите за това точно как сортира **Bubble sort.** Намерете **random number generator** в интернет и генерирайте 100, 1000, 10 000, 100 000 числа. Програмата ви успява ли да се справи с такъв брой елементи?

Normal tasks

4. Прочетете число цяло, положително число **N**, а след това прочетете **N** числа. Напишете функция **int mode(int arrayOfNumbers[], unsigned int size),** който намира модата на редица от числа. Може да считате че въведените числа, ще имат само 1 мода.

Вход	Изход
6 1 3 2 1 4 7	1

5. Прочетете число цяло, положително число **N**, а след това прочетете **N** числа. Напишете функция
void modes(int arrayOfNumbers[], unsigned int size, int modes[], unsigned int& modesSize), който намира всички моди на редица от числа. За какво се използват **modes** и **modesSize**?

Вход	Изход
10 1 3 2 1 3 6 7 8 7 0	1 3 7

6. Прочетете число цяло, положително число **N**, а след това прочетете **N** числа. Напишете функция **int median(int arrayOfNumbers[], unsigned int size),** който намира медианата на редица от числа.

Вход	Изход
7 1 5 3 2 4 6 8	4
6 9 8 4 3 1 3	3.5

7. *Прочетете число цяло, положително число **N**, а след това прочетете **N** числа. Напишете функция **void largestPossibleNumber(int arrayOfNumbers[], unsigned int size),** която показва на конзолата най-голямото число, което може да се получи като се долепят елементите на масива един до друг. Как ще сравняваме числата?

Вход	Изход
5 12 7 1 2 3	732121

8. Прочетете число цяло, положително число **N**, а след това прочетете **N** числа. Напишете функция **void moveZeroesToTheEnd(int arrayOfNumbers[], unsigned int size),** която премества всички елементи със стойност 0 в края на масива.

Вход	Изход
7 1 0 3 2 4 0 5	1 3 2 4 5 0 0

9. Напишете функция **bool addElement(int arrayOfNumbers[], unsigned int& size,int newElement),** която добавя елемент в края масив. Функцията да връща **false** при неуспешно добавяне.

10. Напишете функция **bool removeLast(int arrayOfNumbers[], unsigned int& size, int valueToRemove),** която премахва последното срещане на елемент в масив. Функцията да връща **false** при неуспешно премахване.

11. Напишете функция **bool insertAt(int arrayOfNumbers[], unsigned int size, unsigned int indexAt, int newValue),** която добавя елемент със стойност **newValue** на индекс **indexAt**. Функцията да връща **false** при неуспешно добавяне.

12. Напишете функция **int maxSum(int arrayOfNumbers[], unsigned int size),** която намира максималната сума, която може да се образува от 2 числа, които се образуват от елементите в масива. Разликата в броя на цифрите между двете числа трябва да бъде най-много 1.

Вход	Изход
6 1 7 2 4 6 8	862 + 741 = 1503

7 1 7 2 4 6 8 9	9741 + 862 = 10 603

13. Напишете функция **void maxTripletProduct(int arrayOfNumbers[], unsigned int size),** която намира трите елемента на масива с максималното произведение.

Вход	Изход
5 -4 1 -8 9 6	-4 -8 9
5 1 7 2 -2 5	7 2 5