# MMS Star Camp 17

## КОНТРОЛНО №2

## Задача 1 - символни низове:

От стандартния вход се въвежда цяло положително число (което има най-много 500 цифри). Напишете функция, която кодира низа и връща нов символен низ, за който е заделена динамична памет. Кодирането се случва по следния начин:

* Ако една цифра е на четна позиция, тя се замества със съответстващата главна латинска буква. Например:
* 0 се замества с A
* 1 се замества с B
* 2 се замества с C и т.н.
* Ако една цифра е на нечетна позиция, тя се замества със символ, както е зададено в долната таблица:

| **Цифра** | **Символ** |
| --- | --- |
| 0 | ! |
| 1 | # |
| 2 | / |
| 3 | ~ |
| 4 | = |
| 5 | ‘ |
| 6 | \ |
| 7 | > |
| 8 | . |
| 9 | , |

Погрижете се за освобождаването на динамично заделената памет.

Примерен вход:

**10296126782646987676234**

Примерен изход:

**B!C,G#C\H.C\E\J.H\H\C~E**

## Задача 2 - сортиране:

Направете структура **Car**, за която да се съхраняват следните данни:

* model - *C-string - max 20 symbols*
* maxspeed - *uint8\_t*
* price - *double*

Да се направят 6 функции, с помощта на които да се сортира масив от автомобили (сравняващи функции каквито се подават в стандартната **qsort()**):

1. Сравнение по марка - възходящ лексикографски ред;
2. Сравнение по марка - низходящ лексикографски ред;
3. Сравнение по максимална скорост - възходящ ред;
4. Сравнение по максимална скорост - низходящ ред;
5. Сравнение по цена - възходящ ред.
6. Сравнение по цена - низходящ ред.

Да се направи функция, която да връща сравняваща функция, от тези които дефинирахме по-горе.

**int (\*getComparator(int n))(const void\*, const void\*);**

Номерът на функцията **n**, да е между 1 и 6. При число различно от това, да се върне нулев указател.

Да се създаде масив от 10 автомобила. Да се запълни с произволни стойности:

1. Марка - **[A-Z][a-z]{4,10}**
2. Максимална скорост - 100 - 300
3. Цена - 1000.00 - 100 000.00

Потребител да въведе от стандартния вход цяло число между 1 и 6, с което да избира начин на сортиране на масива от автомобили, като използва функцията **getComparator().**

Сортирането да се извърши със стандартната функция **qsort().**

Да се изведе сортираният масив в подходящ формат.

## Задача 3 - command line arguments

Напишете програма, която намира сумата на аргументите, подадени от командния ред.

Програмата може да приема 3 вида аргументи от командния ред:

* целочислени аргументи в десетична бройна система, например 10, -21, 55.
* числа с плаваща запетая, например 12.5, 3.14
* Числа в шестнадесетична бройна система 0xff

Приемете, че:

* Ако аргумент има точка в записа, то той е число с плаваща запетая
* Ако аргумент започва със 0X или 0x, то той е число записано в шестнадесетична бройна система
* В противен случай аргументът е цяло число, записано в десетична бройна система

Използвайте функциите от стандартната библиотека за да си улесните работата по обработването на аргументите от командния ред (**напр. atoi, strtof, sscanf от string.h**).

### Примерно изпълнение:

**$ ./sum\_cli 10 12.5 0xff -21 55 0XA 3.14**

**Sum is 324.64**

### Пояснение:

**10 + 12.5 +** 0xff**(255) - 21 + 55 +** 0XA**(10) + 3.14 = 324.64**