Задача 1 (2 т.). Куриерска фирма иска да е сигурна, че доставчиците не крадат и не подменят артикулите в пратките. За целта тя въвежда код, с който да проверява дали съдържанието на пратката е променено. Кодът се генерира на база на **сумата** от контролните числа на всички артикули, които пратката съдържа.

За всеки артикул контролното число се изчислява с формулата - сумата от символите в името на артикула (за сумата да се използва съответния номер в ASCII таблицата) умножена по $31^{\text{пореден индекс на артикул}}$.

За решението на задачата всяка пратка следва да се представи като низ с имена на артикули разделени с **интервал**.

Да се напишат следните функции:

- `inputShipment` – въвежда данните за пратка от стандартния вход (“*mug snow-man*”). Приема се, че дължината на низа се въвежда от клавиатурата, кодът не може да съдържа повече от 10 цифри.
- `packingShipment` – опакова пратката като генерира код за проверка и го добавя в края на низа за дадената пратка. Кодът да е отделен с интервал (“*mug snow-man 26152*”).
- `inspectShipment` – приема дадена опакована пратка и проверява дали пратката е била променяна. (“*mug snow-man 26152*” → не е променена
  “*mug snow-woman 26152*” → променена е).

Обяснение на примера:
пратка – “mug snow-man”
контролно число на първи артикул: m(109) + u(117) + g(103) = 329 * 31^0 = 329
контролно число на втори артикул: s(115) + n(110) + o(111) + w(119) + ‘-’(45) + m(109) + a(97) + n(110) = 833 * 31^1 = 25823
Код на пратката: 329 + 25823 = 26152

Задача 2. (1 т.) Да се напише програма, която въвежда правоъгълна матрица от думи с размерност n x m (1 <= n, m <= 10) и намера изречението, което се образува след последователното конкатениране на думите, обхождайки таблицата по редове, започвайки от долния ред и отдясно, наляво.

Задача 3. (1 т.) Да се напише булева функция, която **по указател** към началото на низ от малки латински букви и цифри, връща дали има число в низа. Функцията да записва И указателите към началото и края на най-голямото число. Ако има две еднакви числа, указателите да сочат към първото от тях. Стойностите на указателите да остават видими и след края на изпълнение на функцията.

Пример:
“p4fd456bc23ll” → връща “истина”, указател към първата цифра ‘4’ и към последната ‘6’.

Задача за анализ.
Да се определи какъв е изходът от изпълнението на програмата.
Да се опишат какво правят реализираните функции.

Да се допишат празните места, така че кодът да изпълнява поставената задача.

Основна постановка на задачата: Аня и Ванката искат да засадят K елхи край една улица. На тази улица са предвидени N (1 <= N <= 1024) възможни места за засаждане. Те искат да засадят елхите така, че най-малкото разстояние между две съседни дървета да е възможно най-голямо.
Пример:
Вход: N = 5; K = 2
Разстояния от началото на улицата до всяко от местата: 5, 8, 12, 32, 1.

Вход: N = 7; M = 3
Разстояния от началото на улицата до всяко от местата: 1, 15, 35, 10, 59, 60, 28.

```
#include <iostream>
using namespace std;

// 1.
bool isPossible(size_t* arr, size_t size, size_t trees, size_t x) {
    size_t last = 1.1_________________;
    size_t counter = 1.2_________________;
    for (size_t i = 1; i < size; i++) {
        if (arr[i] - arr[last] >= x) {
            last = i;
            counter++;
        }
    }

    return 1.3_________________;
}

// 2.
size_t findDistance(size_t* arr, size_t trees, size_t size) {
    size_t l = 2.1_________________;
    size_t r = 2.2_________________;
    size_t distance = 2.3_________________;
    while (l <= r) {
        2.4_________________;
        if (isPossible(2.5_________________)) {
            2.6_________________;
            distance = 2.7_________________;
        }
        else
        2.8_________________;
    }
    return distance;
}


void sortArray(size_t* arr, size_t size) {
    for (size_t i = 0; i < size - 1; ++i) {
        size_t minIdx = i;

        for (size_t j = i + 1; j < size; ++j) {
            if (arr[j] < arr[minIdx]) {
                minIdx = j;
```

```
            }
        }

        size_t temp = arr[i];
        arr[i] = arr[minIdx];
        arr[minIdx] = temp;
    }
}

int main() {
    size_t n = 0;
    size_t k = 0;
    cin >> n >> k;
    size_t* arr = new size_t[n];
    for (size_t i = 0; i < n; ++i) {
        cin >> arr[i];
    }
    3.1_____________________________
    3.2_____________________________
    delete[] arr;
    return 0;
}
```