TASK 1 - Sort

```
void sort(std::vector<int>& nums);
```

You are given an list of numbers and your task is to sort them in increasing order.

The method will not return a new vector or an array, it modifies the original vector passed by reference.

There is a constraint on the numbers in the list. They are only limited to digits, ie. the range [0..9].
There are sorting algorithms that run in $O(n^2)$ or $O(nlg(n))$, we are looking for an $O(n)$ solution for this task.

an example:

```
Given a list of shuffled numbers as { 1 1 2 3 1 0 9 8 5 7 5 }
      modify it so that it becomes { 0 1 1 1 2 3 5 5 7 8 9 }
```

TASK 2 - Odd Sum

```
int odd(std::vector<int>& nums);
```

You are given an list of **non-negative** numbers in range [0,1000) and your task is find the maximum odd sum of a continuous subsequence of the array. An odd sum is a sum of numbers that is not even. The numbers in the summed sequence may contain even or odd numbers but their sum must be odd.

It is possible to check every continuous subsequence and find maximum odd sum in $O(n^2)$, we are looking for an $O(n)$ solution for this task.

an example:

```
Given a list of numbers as { 6 4 2 1 1 2 3 1 0 9 8 5 4 2 }
selecting this subsequence { 6 4 2 1 1 2 3 1 0 9 8 - - - }
gives the maximum odd sum of 37
```

TASK 3 - Search

```
int search(std::vector<int>& nums, int num);
```

You are given a vector of numbers and a number. Your task is to determine whether *num* is in the vector and return the index of the number in the vector. If the given number cannot be found, return -1. The numbers are *integers* in range [-1000000, 2000000).

The numbers have some specific properties. The vector contains **unique** elements without any duplicates. Additionally the elements are **sorted** in increasing order to a point, then they are still sorted but in decreasing order.

A trivial linear search implementation will return the index in **O(n)** time, n being the size of the *nums* vector.

We are looking for **O(lg(n))** solutions for this task.

an example:

```
Given this array of numbers { 0 1 3 4 7 8 6 5 2 }

                with indices { 0 1 2 3 4 5 6 7 8 }

Then, the index of 5 is 7, the index of 3 is 2.
```