3. 628. Maximum Product of Three Numbers

   Given an integer array nums, find three numbers whose product is maximum and return the maximum product.

```
//Time Complexity : O(n)

//Space Complexity: O(1)

class Solution {

public:

    int maximumProduct(vector<int>& nums) {



        int n=nums.size();

        sort(nums.begin(), nums.end());

        int result=max((nums[0]*nums[1]*nums[n-1]),(nums[n-1]*nums[n-2]*nums[n-3]));

        return result;

    }

};
```

4. 704. Binary Search

Given an array of integers nums which is sorted in ascending order, and an integer target, write a function to search target in nums. If target exists, then return its index. Otherwise, return -1.

You must write an algorithm with O(log n) runtime complexity.

```
//Time Complexity: O(log n)

//Space Complexity: O(1)

class Solution {

public:

    int search(vector<int>& nums, int target) {

        int low=0;

        int high=nums.size()-1;

        while(low<=high)

        {

            int mid=low+(high-low)/2;

            if(target<nums[mid])

            {

                high=mid-1;

            }

            else if(target>nums[mid])

            {

                low=mid+1;

            }

            else

            {
                return mid;
            }
        }
        return -1;
    }
};
```

5. 605. Can Place Flowers

You have a long flowerbed in which some of the plots are planted, and some are not. However, flowers cannot be planted in adjacent plots.

Given an integer array flowerbed containing 0's and 1's, where 0 means empty and 1 means not empty, and an integer n, return true if n new flowers can be planted in the flowerbed without violating the no-adjacent-flowers rule and false otherwise.

```
//Time Complexity: O(n)

//Space Complexity: O(1)

class Solution {

public:

    bool canPlaceFlowers(vector<int>& flowerbed, int n) {

        if(n==0)

        {

            return true;

        }

        for(int i=0;i<flowerbed.size();i++)

        {

            if(flowerbed[i]==0 && (i==0 || flowerbed[i-1]==0) &&
(i==flowerbed.size()-1 || flowerbed[i+1]==0))

            {

            flowerbed[i]=1;

            n--;

              if(n==0)

              {

                  return true;
```

```
               }

            }

        }

        return false;

       }

};
```

6. 896. Monotonic Array

An array is monotonic if it is either monotone increasing or monotone decreasing.

An array nums is monotone increasing if for all i <= j, nums[i] <= nums[j]. An array nums is monotone decreasing if for all i <= j, nums[i] >= nums[j].

Given an integer array nums, return true if the given array is monotonic, or false otherwise.

Code:

```
//Time Complexity: O(n)
//Space Complexity: O(1)
class Solution {
public:
    bool isMonotonic(vector<int>& nums) {
        bool inc=true;
        bool dec=true;
        for(int i=0;i<nums.size()-1;i++)
        {
            if(nums[i]>nums[i+1])
            {
                inc=false;
            }
            if(nums[i]<nums[i+1])
            {
                dec=false;
            }
            if(inc==false && dec==false)
            {
                return false;
            }

        }
        return true;
    }
};
```