Arrays
(Assignment Solutions)

Question 1 : Sorting is used in this solution but more efficient solutions exist using other data structures we haven't studied yet.

```
bool containsDuplicate(vector<int>& nums) {
       sort(nums.begin(), nums.end());

       for(int i=1; i<nums.size(); i++) {
           if(nums[i-1] == nums[i]) {
               return true;
           }
       }

       return false;
   }
```

Question 2 : This problem will be solved using the binary search approach. Infact, whenever the Qs asks us to solve it with O(nlogn) complexity, it is sometimes a hint towards some sort of a binary approach.

```
int search(vector<int>& nums, int target) {
       int low = 0, high = nums.size()-1;

       while (low <= high) {
           int mid = (low + high) / 2;

           if (nums[mid] == target) {
               return mid;
           }

           if (nums[low] <= nums[mid]) {
               if (nums[low] <= target && target < nums[mid]) {
                   high = mid - 1;
```

```
                } else {
                    low = mid + 1;
                }
            } else {
                if (nums[mid] < target && target <= nums[high]) {
                    low = mid + 1;
                } else {
                    high = mid - 1;
                }
            }
        }

        return -1;
    }
```

Question 3 :

```
int maxProduct(vector<int>& nums) {
        int maxTillNow = nums[0];
        int minTillNow = nums[0];
        int ans = maxTillNow;

        for (int i=1; i<nums.size(); i++) {
            int curr = nums[i];

                      int  tempMaxTillNow  =  max(curr,  max(maxTillNow*curr,
minTillNow*curr));
            minTillNow = min(curr, min(maxTillNow*curr, minTillNow*curr));
            maxTillNow = tempMaxTillNow;

            ans = max(maxTillNow, ans);
        }

        return ans;
    }
```