**Assignment Questions 2 Solutions:**

Qs.1.

class Solution:

def arrayPairSum(self, nums):

"""

:type nums: List[int]

:rtype: int

"""

# approach: sort list, take smaller ones and sum them up

nums.sort()

return sum(nums[::2])

Qs.2.

class Solution:

def distributeCandies(self, candyType: List[int]) -> int:

l = len(candyType)

eat = l//2

dis\_candyType = set(candyType)

if eat <= len(dis\_candyType):

return eat

elif eat > len(dis\_candyType):

return len(dis\_candyType)

Qs.3.

class Solution:

def findLHS(self, nums: List[int]) -> int:

freq = {}

for num in nums:

freq[num] = freq.get(num, 0) + 1

max\_length = 0

for num in freq:

if num + 1 in freq:

max\_length = max(max\_length, freq[num] + freq[num + 1])

return max\_length

Qs.4.

class Solution(object):

def canPlaceFlowers(self, flowerbed, n):

count = 1

beds = 0

for bed in flowerbed:

if bed:

count = 0

else:

count += 1

if count == 3:

beds += 1

count = 1

if not flowerbed[-1]:

count += 1

if count == 3: beds += 1

return beds >= n

Qs.5.

class Solution {

public int maximumProduct(int[] nums) {

Arrays.sort(nums);

int case1 = nums[0]\*nums[1]\*nums[nums.length-1];

int case2 = nums[nums.length-1]\*nums[nums.length-2]\*nums[nums.length-3];

int maxProduct = Integer.max(case1, case2);

return maxProduct;

}

}

Qs.6

class Solution {

public int search(int[] nums, int target) {

int low = 0, high = nums.length -1, mid;

while(low <= high){

mid = (low + high)/2;

if (nums[mid] == target)

return mid;

if(target < nums[mid]){

high = mid-1;

}else

low = mid+1;

}

return -1;

}

}

Qs.7

class Solution {

public:

bool montoneincrease(vector<int> &nums)

{

for(int i=0;i<nums.size()-1;i++)

{

if(nums[i]>nums[i+1]) return false;

}

return true;

}

bool montonedecrease(vector<int> &nums)

{

for(int i=0;i<nums.size()-1;i++)

{

if(nums[i]<nums[i+1]) return false;

}

return true;

}

bool isMonotonic(vector<int>& nums) {

if(nums.size()==1) return true;

if(nums[0]>nums[1])

{

return montonedecrease(nums);

}

else if(nums[0]<nums[1])

{

return montoneincrease(nums);

}

else

for(int i=0;i<nums.size()-1;i++)

{

if(nums[i]==nums[i+1])

continue;

else if(nums[i]>nums[i+1]) return montonedecrease(nums);

else return montoneincrease(nums);

}

return true;

}

};

Qs.8.

class Solution {

public int smallestRangeI(int[] nums, int k) {

//find max and min element

int max, min;

max=min=nums[0];

for(int i=0;i<nums.length;i++){

max=Math.max(max,nums[i]);

min=Math.min(min,nums[i]);

}

// how will be the difference will be minimum

// when we max=max-k and min=min+k;

int diff= (max-k)-(min+k);

return (diff>0) ? diff : 0;

}

}