**Assignment Questions 4 Solutions:**

Qs.1.

#include <bits/stdc++.h>

using namespace std;

void findCommon(int ar1[], int ar2[], int ar3[], int n1,

int n2, int n3)

{

int i = 0, j = 0, k = 0;

while (i < n1 && j < n2 && k < n3) {

if (ar1[i] == ar2[j] && ar2[j] == ar3[k]) {

cout << ar1[i] << " ";

i++;

j++;

k++;

}

else if (ar1[i] < ar2[j])

i++;

// y < z

else if (ar2[j] < ar3[k])

j++;

// We reach here when x > y and z < y, i.e., z is

// smallest

else

k++;

}

}

int main()

{

int ar1[] = { 1, 5, 10, 20, 40, 80 };

int ar2[] = { 6, 7, 20, 80, 100 };

int ar3[] = { 3, 4, 15, 20, 30, 70, 80, 120 };

int n1 = sizeof(ar1) / sizeof(ar1[0]);

int n2 = sizeof(ar2) / sizeof(ar2[0]);

int n3 = sizeof(ar3) / sizeof(ar3[0]);

cout << "Common Elements are ";

findCommon(ar1, ar2, ar3, n1, n2, n3);

return 0;

}

Qs.2.

vector<int> getElementsOnlyInFirstList(vector<int>& nums1, vector<int>& nums2) {

unordered\_set<int> onlyInNums1;

// Iterate over each element in the list nums1.

for (int num : nums1) {

bool existInNums2 = false;

// Check if num is present in the second arg nums2.

for (int x : nums2) {

if (x == num) {

existInNums2 = true;

break;

}

}

if (!existInNums2) {

onlyInNums1.insert(num);

}

}

// Convert to vector.

return vector<int> (onlyInNums1.begin(), onlyInNums1.end());

}

Qs.3.

vector<vector<int>> transpose(vector<vector<int>>& matrix) {

vector<vector<int>> ans;

cout<<matrix.size();

cout<<matrix[0].size();

for(int i=0;i<matrix[0].size();i++)

{

vector<int>res;

for(int j=0;j<matrix.size();j++)

{

res.push\_back(matrix[j][i]);

}

ans.push\_back(res);

}

return ans;

}

Qs.4.

public static class Solution1 {

public int arrayPairSum(int[] nums) {

Arrays.sort(nums);

int total = 0;

for (int i = 0; i < nums.length; i += 2) {

total += nums[i];

}

return total;

}

}

Qs.5.

class Solution {

public int arrangeCoins(int n) {

return (int)(Math.sqrt(2 \* (long)n + 0.25) - 0.5);

}

}

Qs.6.

vector<int> sortedSquares(vector<int>& nums) {

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

{

nums[i]=nums[i]\*nums[i];

}

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

return nums;

}

Qs.7.

class Solution:

def maxCount(self, m: int, n: int, ops: List[List[int]]) -> int:

length = len(ops)

if length == 0:

return m\*n

result = [ops[0][0] , ops[0][1]]

for i in range(1,length):

result[0] = min(result[0] , ops[i][0])

result[1] = min(result[1] , ops[i][1])

return result[0]\*result[1]

Qs.8.

class Solution {

public:

vector<int> shuffle(vector<int>& nums, int n) {

vector<int>ans;

int i = 0, j=n;

while(i<n && j<(2\*n))

{

ans.push\_back(nums[i]);

ans.push\_back(nums[j]);

i++;

j++;

}

return ans;

}

};