ASSIGNMENT 4

1.

```
def arraysIntersection(arr1, arr2, arr3):
    i = j = k = 0
    result = []
    while i < len(arr1) and j < len(arr2) and k < len(arr3):
        if arr1[i] == arr2[j] == arr3[k]:
            result.append(arr1[i])
            i += 1
            j += 1
            k += 1
        elif arr1[i] <= arr2[j] and arr1[i] <= arr3[k]:
            i += 1
        elif arr2[j] <= arr1[i] and arr2[j] <= arr3[k]:
            j += 1
        else:
            k += 1
    return result
```

2.

```
def arraysIntersection(nums1, nums2):
    count1 = collections.Counter(nums1)
    count2 = collections.Counter(nums2)
    result = []
    for num in nums1:
        if num in count1 and num in count2:
            result.append(num)
            count1[num] -= 1
            count2[num] -= 1
            if count1[num] == 0:
                del count1[num]
            if count2[num] == 0:
```

```
            del count2[num]
    return result
```

3.

```
def transpose(matrix):
    m, n = len(matrix), len(matrix[0])
    result = [[0] * m for _ in range(n)]
    for i in range(m):
        for j in range(n):
            result[j][i] = matrix[i][j]
    return result
```

4.

```
def arrayPairSum(nums):
    nums.sort()
    n = len(nums)
    result = 0
    for i in range(0, n, 2):
        result += nums[i]
    return result
```

5.

```
def arrangeCoins(n):
    left, right = 1, n
    while left <= right:
        mid = left + (right - left) // 2
        total = mid * (mid + 1) // 2
        if total == n:
            return mid
        elif total < n:
            left = mid + 1
        else:
            right = mid - 1
    return right
```

6.

```
def sortedSquares(nums):
    n = len(nums)
    result = [0] * n
    left, right = 0, n - 1
    for i in range(n - 1, -1, -1):
        if abs(nums[left]) > abs(nums[right]):
            result[i] = nums[left] * nums[left]
            left += 1
        else:
            result[i] = nums[right] * nums[right]
            right -= 1
    return result
```

7.

```
def maxCount(m, n, ops):
    min_row = m
    min_col = n
    for op in ops:
        min_row = min(min_row, op[0])
        min_col = min(min_col, op[1])
    return min_row * min_col
```

8.

```
def shuffle(nums, n):
    result = []
    for i in range(n):
        result.append(nums[i])
        result.append(nums[i + n])
    return result
```