ASSIGNMENT 5

1.

```
def convert_to_2d(original, m, n):
    if m * n != len(original):
        return []

    result = []
    for i in range(m):
        row = original[i * n : (i + 1) * n]
        result.append(row)

    return result
original = [1, 2, 3, 4]
m = 2
n = 2
result = convert_to_2d(original, m, n)
print(result)
```

2.

```
def count_complete_rows(n):
    k = 0
    while (k * (k + 1)) // 2 <= n:
        k += 1
    return k - 1
n = 5
result = count_complete_rows(n)
print(result)
```

3.

```
def sorted_squares(nums):
    result = []
    for num in nums:
        result.append(num * num)
```

```
    result.sort()
    return result
nums = [-4, -1, 0, 3, 10]
result = sorted_squares(nums)
print(result)
```

4.

```
def find_distinct_elements(nums1, nums2):
    set1 = set(nums1)
    set2 = set(nums2)
    distinct_nums1 = list(set1 - set2)
    distinct_nums2 = list(set2 - set1)
    return [distinct_nums1, distinct_nums2]
nums1 = [1, 2, 3]
nums2 = [2, 4, 6]
result = find_distinct_elements(nums1, nums2)
print(result)
```

5.

```
def distance_value(arr1, arr2, d):
    count = 0
    for num1 in arr1:
        for num2 in arr2:
            if abs(num1 - num2) <= d:
                break
        else:
            count += 1
    return count
arr1 = [4, 5, 8]
arr2 = [10, 9, 1, 8]
d = 2
result = distance_value(arr1, arr2, d)
print(result)
```

6.

```
def find_duplicates(nums):
    frequency = {}
    result = []
    for num in nums:
        if num in frequency:
            frequency[num] += 1
        else:
            frequency[num] = 1
        if frequency[num] == 2:
            result.append(num)
    return result
nums = [4, 3, 2, 7, 8, 2, 3, 1]
result = find_duplicates(nums)
print(result)
```

7.

```
def find_minimum(nums):
    left = 0
    right = len(nums) - 1

    while left < right:
        mid = (left + right) // 2

        if nums[mid] > nums[right]:
            left = mid + 1
        else:
            right = mid

    return nums[left]
nums = [3, 4, 5, 1, 2]
result = find_minimum(nums)
```

```
print(result)
```

8.

```
from collections import defaultdict

def find_original(changed):
    if len(changed) % 2 != 0:
        return []

    frequency = defaultdict(int)
    for num in changed:
        frequency[num] += 1

    original = []
    for num in frequency:
        if frequency[num] > 0:
            original.extend([num] * (frequency[num] // 2))

    return original
changed = [1, 3, 4, 2, 6, 8]
result = find_original(changed)
print(result)
```