1. Given a sorted array of positive and negative numbers. You have to Square it and sort it.

Constraint : Time complexity O(n)

Example:

Input: [-12, -8 , -7, -5, 2, 4, 5, 11, 15]

Output : [4, 16, 25, 25, 49, 56, 121, 144, 225]

```
def square_and_sorted_arr(arr):
    squared_arr = []
    for i in range(len(arr)):
        squared_arr.append(arr[i]**2)

    return sorted(squared_arr)


arr = [-12, -8, -7, -5, 2, 4, 5, 11, 15]
print(square_and_sorted_arr(arr))
```

2. Design an immutable class with following attributes

String name;

String Id,

Date dateOfJoining

List<Address> addresses;

```
from dataclasses import dataclass, field
from datetime import date
from typing import List
import copy

@dataclass(frozen=True)
```

```
class Address:
    street: str
    city: str
    state: str
    zip_code: str

@dataclass(frozen=True)
class Employee:
    name: str
    Id: str
    dateOfJoining: date
    addresses: List[Address] = field(default_factory=list)

    def __post_init__(self):

        object.__setattr__(self, "addresses", tuple(copy.deepcopy(self.addresses)))

address1 = Address("123 Main St", "New York", "NY", "10001")
address2 = Address("456 Maple Rd", "Los Angeles", "CA", "90001")

employee = Employee(name="John Doe", Id="E12345", dateOfJoining=date(2020, 5,
15), addresses=[address1, address2])

print(employee)
```

3. Given an array of Red Green Blue balls.You have to sort it.

Constraint : Time complexity O(n)

Constraint : Space complexity O(1)

Example:

Input: [R, G, B, G, G, R, B, B, G]

Output : [B,B,B,G,G,G,G,R, R]

```
def sort_balls(arr):
    low, mid, high = 0, 0, len(arr) - 1

    while mid <= high:
        if arr[mid] == 'B':
            arr[low], arr[mid] = arr[mid], arr[low]
            low += 1
            mid += 1
        elif arr[mid] == 'G':
            mid += 1
        else:
            arr[mid], arr[high] = arr[high], arr[mid]
            high -= 1

    return arr

balls = ['R', 'G', 'B', 'G', 'G', 'R', 'B', 'B', 'G']
sorted_balls = sort_balls(balls)
print(sorted_balls)
```

4. We are given two arrays that represent the arrival and departure times of trains, the task is to find the minimum number of platforms required so that no train waits.

Examples:

Input: arr[] = {9:00, 9:40, 9:50, 11:00, 15:00, 18:00}, dep[] = {9:10, 12:00, 11:20, 11:30,

19:00, 20:00}

Output: 3

Explanation: There are at-most three trains at a time (time between 9:40 to 12:00)

Input: arr[] = {9:00, 9:40}, dep[] = {9:10, 12:00}

Output: 1

Explanation: Only one platform is needed.

```
def find_min_platforms(arr, dep):
    n = len(arr)



    arr = sorted([int(t.split(":")[0]) * 60 + int(t.split(":")[1]) for t in arr])
    dep = sorted([int(t.split(":")[0]) * 60 + int(t.split(":")[1]) for t in dep])

    i, j = 0, 0
    platforms_needed = 0
    max_platforms = 0

    while i < n and j < n:
        if arr[i] <= dep[j]:
            platforms_needed += 1
            max_platforms = max(max_platforms, platforms_needed)
            i += 1
        else:
            platforms_needed -= 1
            j += 1

    return max_platforms


arr = ["9:00", "9:40", "9:50", "11:00", "15:00", "18:00"]
dep = ["9:10", "12:00", "11:20", "11:30", "19:00", "20:00"]
print(find_min_platforms(arr, dep))

arr2 = ["9:00", "9:40"]
dep2 = ["9:10", "12:00"]
print(find_min_platforms(arr2, dep2))
```

5. Sort hashmap by value.

Example:

Input: Map: {101=John Doe, 102=Jane Smith, 103=Peter Johnson}

output: Map: {102=Jane Smith, 101=John Doe, 103=Peter Johnson}

```
def sort_dict_by_value(input_dict):
    return dict(sorted(input_dict.items(), key=lambda item: item[1]))

my_map= {101: "John Doe", 102: "Jane Smith", 103: "Peter Johnson"}
sorted_map = sort_dict_by_value(my_map)
print(sorted_map)
```