Question 1

g) Exercise 2

```
def count_significant_inversions(arr):
    def merge_sort(low, high):
        if high - low <= 1:
            return 0
        mid = (low + high) // 2
        count = merge_sort(low, mid) + merge_sort(mid, high)
        j = mid
        # Count significant inversions
        for i in range(low, mid):
            while j < high and arr[i] > 2 * arr[j]:
                j += 1
            count += j - mid
        # Standard merge step
        arr[low:high] = sorted(arr[low:high])
        return count
    return merge_sort(0, len(arr))
```

h)

Exercise 1

```
def find_median_sorted(A, B):
    n = len(A)
    low, high = 0, n
    while low <= high:
        i = (low + high) // 2
        j = n - i
```

```
if i > 0 and A[i-1] > B[j]:
    high = i - 1
elif i < n and B[j-1] > A[i]:
    low = i + 1
else:
    left = max(A[i-1] if i > 0 else float('-inf'),
               B[j-1] if j > 0 else float('-inf'))
    right = min(A[i] if i < n else float('inf'),
                B[j] if j < n else float('inf'))
    return (left + right) / 2
```

Exercise 2

```
def count_significant_inversions(arr):
    def merge_sort(low, high):
        if high - low <= 1:
            return 0
        mid = (low + high) // 2
        count = merge_sort(low, mid) + merge_sort(mid, high)

        j = mid
        for i in range(low, mid):
            while j < high and arr[i] > 2 * arr[j]:
                j += 1
            count += j - mid

        # Merge step (sort the subarray)
        arr[low:high] = sorted(arr[low:high])
        return count

    return merge_sort(0, len(arr))
```

Exercise 6

```
def subarray_sums(A):
    n = len(A)
    prefix = [0]
    for x in A:
        prefix.append(prefix[-1] + x)
    B = [[0]*n for _ in range(n)]
    for i in range(n):
```

```
    for j in range(i+1, n):
        B[i][j] = prefix[j+1] - prefix[i]
return B
```