1.
Equilibrium Index Element

https://www.interviewbit.com/problems/balance-array/

```
import java.util.Arrays;

public class BalanceArray {

  public static int balance_array(int[] A) {
    int[] prefix_sum = new int[A.length];
    prefix_sum[0] = A[0];
    for (int i = 1; i < A.length; i++) {
      prefix_sum[i] = prefix_sum[i - 1] + A[i];
    }
```

2.

Pick from both sides!
https://www.interviewbit.com/problems/pick-from-both-sides/

```java
import java.util.*;

public class PickFromBothSides {

    public static int pickFromBothSides(int[] arr, int b) {
        int n = arr.length;
        int[] prefixSum = new int[n];
        prefixSum[0] = arr[0];
        for (int i = 1; i < n; i++) {
            prefixSum[i] = prefixSum[i - 1] + arr[i];
        }

        int maxSum = Integer.MIN_VALUE;
        for (int i = 0; i < n; i++) {
            int j = i + b - 1;
            if (j < n) {
                maxSum = Math.max(maxSum,
prefixSum[j] - (i == 0 ? 0 : prefixSum[i - 1]));
            }
        }
        return maxSum;
    }

    public static void main(String[] args) {
        int[] arr = {5, -2, 3, 1, 2};
        int b = 3;

        int maxSum = pickFromBothSides(arr, b);
        System.out.println(maxSum);
    }
}
```

```
    int even_sum = 0;
    int odd_sum = 0;
    int count = 0;

    for (int i = 0; i < A.length; i++) {
      if (i % 2 == 0) {
        even_sum += A[i];
      } else {
        odd_sum += A[i];
      }
```

```
        if (even_sum == odd_sum) {
          count++;
        }
      }

      return count;
    }

    public static void main(String[] args) {
      int[] A = {2, 1, 6, 4};

      int count = balance_array(A);

      System.out.println(count);
    }
  }
```

3.
https://leetcode.com/problems/minimum-operations-to-make-array-equal/

```
import java.util.*;
```

```
public class MinimumOperationsToMakeArrayEqual
{

    public static int minOperations(int[] arr) {
        int n = arr.length;
        int[] prefixSum = new int[n];
        for (int i = 1; i < n; i++) {
            prefixSum[i] = prefixSum[i - 1] + arr[i];
        }

        int minOps = Integer.MAX_VALUE;
        for (int i = 0; i < n; i++) {
            for (int j = i; j < n; j++) {
                int target = (j - i) * arr[i];
                if (prefixSum[j] - prefixSum[i] == target) {
                    minOps = Math.min(minOps, j - i);
                }
            }
        }

        return minOps;
    }

    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 4, 5};
        System.out.println(minOperations(arr));
    }
}
```

4.
303. Range Sum Query - Immutable : https://leetcode.com/problems/range-sum-query-immutable/'

```
class NumArray {

  private int[] prefixSums;

  public NumArray(int[] nums) {
    int n = nums.length;
    prefixSums = new int[n + 1];
    prefixSums[0] = 0;
    for (int i = 1; i <= n; i++) {
      prefixSums[i] = prefixSums[i - 1] + nums[i - 1];
    }
  }

  public int sumRange(int i, int j) {
    return prefixSums[j + 1] - prefixSums[i];
  }
}
```

5.
Equilibrium Point : https://practice.geeksforgeeks.org/problems/equilibrium-point-1587115620/1?utm_source=gfg&utm_medium=article&utm_campaign=bottom_sticky_on_article

```
public class PrefixArrayIndex {

    public static int findIndex(int[] prefixArray, int
value) {
        int n = prefixArray.length;
    int low = 0;
    int high = n - 1;

    while (low <= high) {
        int mid = (low + high) / 2;

        if (prefixArray[mid] == value) {
            return mid;
        } else if (prefixArray[mid] < value) {
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }

    return -1;
}

public static void main(String[] args) {
    int[] prefixArray = {1, 2, 3, 4, 5, 6, 7};
    int value = 5;

    int index = findIndex(prefixArray, value);

    if (index != -1) {
            System.out.println("The index of " + value + "
in the prefix array is " + index);
        } else {
            System.out.println("The value " + value + " is
not present in the prefix array");
        }
    }
}
```

6.
Product of Array Except Self : https://leetcode.com/problems/product-of-array-except-self/description/

```
public class PrefixArrayIndex {

    public static int[] prefixArrayIndex(int[] nums) {
        int n = nums.length;
        int[] answer = new int[n];

        // Calculate the prefix products.
    int[] prefixProduct = new int[n];
    prefixProduct[0] = nums[0];
    for (int i = 1; i < n; i++) {
        prefixProduct[i] = prefixProduct[i - 1] * nums[i];
    }

    // Calculate the answer array.
    for (int i = 0; i < n; i++) {
        answer[i] = prefixProduct[i];
        for (int j = 0; j < i; j++) {
            answer[i] /= nums[j];
        }
    }

    return answer;
}

public static void main(String[] args) {
    int[] nums = {1, 2, 3, 4, 5};
    int[] answer = prefixArrayIndex(nums);

    for (int i = 0; i < answer.length; i++) {
        System.out.print(answer[i] + " ");
    }
```

```
        System.out.println();
    }
}
```