Java

```java
/**
 * // This is ArrayReader's API interface.
 * // You should not implement it, or speculate about its implementation
 * interface ArrayReader {
 *     // Compares the sum of arr[l..r] with the sum of arr[x..y]
 *     // return 1 if sum(arr[l..r]) > sum(arr[x..y])
 *     // return 0 if sum(arr[l..r]) == sum(arr[x..y])
 *     // return -1 if sum(arr[l..r]) < sum(arr[x..y])
 *     public int compareSub(int l, int r, int x, int y) {}
 *
 *     // Returns the length of the array
 *     public int length() {}
 * }
 */

class Solution {
    public int getIndex(ArrayReader reader) {

    }
}
```

JavaScript

```javascript
/**
 * // This is the ArrayReader's API interface.
 * // You should not implement it, or speculate about its implementation
 * function ArrayReader() {
 *     // Compares the sum of arr[l..r] with the sum of arr[x..y]
 *     // return 1 if sum(arr[l..r]) > sum(arr[x..y])
```

```
 *     // return 0 if sum(arr[l..r]) == sum(arr[x..y])
 *     // return -1 if sum(arr[l..r]) < sum(arr[x..y])
 *     @param {number} l, r, x, y
 *     @return {number}
 *     this.compareSub = function(l, r, x, y) {
 *         ...
 *     };
 *
 *     // Returns the length of the array
 *     @return {number}
 *     this.length = function() {
 *         ...
 *     };
 * };
 */

/**
 * @param {ArrayReader} reader
 * @return {number}
 */
var getIndex = function(reader) {

};
```

TypeScript

```
/**
 * // This is the ArrayReader's API interface.
 * // You should not implement it, or speculate about its implementation
 * class ArrayReader {
 *     // Compares the sum of arr[l..r] with the sum of arr[x..y]
```

```
 *     // return 1 if sum(arr[l..r]) > sum(arr[x..y])
 *     // return 0 if sum(arr[l..r]) == sum(arr[x..y])
 *     // return -1 if sum(arr[l..r]) < sum(arr[x..y])
 *     compareSub(l: number, r: number, x: number, y: number): number { };
 *
 *     // Returns the length of the array
 *     length(): number { };
 * };
 */

function getIndex(reader: ArrayReader): number {

};
```

C++

```
/**
 * // This is the ArrayReader's API interface.
 * // You should not implement it, or speculate about its implementation
 * class ArrayReader {
 *   public:
 *     // Compares the sum of arr[l..r] with the sum of arr[x..y]
 *     // return 1 if sum(arr[l..r]) > sum(arr[x..y])
 *     // return 0 if sum(arr[l..r]) == sum(arr[x..y])
 *     // return -1 if sum(arr[l..r]) < sum(arr[x..y])
 *     int compareSub(int l, int r, int x, int y);
 *
 *     // Returns the length of the array
 *     int length();
 * };
 */
```

```
class Solution {
public:
    int getIndex(ArrayReader &reader) {

    }
};
```

C#

```
/**
 * // This is ArrayReader's API interface.
 * // You should not implement it, or speculate about its implementation
 * class ArrayReader {
 *     // Compares the sum of arr[l..r] with the sum of arr[x..y]
 *     // return 1 if sum(arr[l..r]) > sum(arr[x..y])
 *     // return 0 if sum(arr[l..r]) == sum(arr[x..y])
 *     // return -1 if sum(arr[l..r]) < sum(arr[x..y])
 *     public int CompareSub(int l, int r, int x, int y) {}
 *
 *     // Returns the length of the array
 *     public int Length() {}
 * }
 */

class Solution {
    public int GetIndex(ArrayReader reader) {

    }
}
```

Kotlin

```
/**
 * // This is ArrayReader's API interface.
 * // You should not implement it, or speculate about its implementation
 * interface ArrayReader {
 *     // Compares the sum of arr[l..r] with the sum of arr[x..y]
 *     // return 1 if sum(arr[l..r]) > sum(arr[x..y])
 *     // return 0 if sum(arr[l..r]) == sum(arr[x..y])
 *     // return -1 if sum(arr[l..r]) < sum(arr[x..y])
 *     fun compareSub(l: Int, r: Int, x: Int, y: Int): Int {}
 *
 *     // Returns the length of the array
 *     fun length(): Int {}
 * }
 */

class Solution {
    fun getIndex(reader: ArrayReader): Int {

    }
}
```

Go

```
/**
 * // This is the ArrayReader's API interface.
 * // You should not implement it, or speculate about its implementation
 * type ArrayReader struct {
 * }
```

```
 * // Compares the sum of arr[l..r] with the sum of arr[x..y]
 * // return 1 if sum(arr[l..r]) > sum(arr[x..y])
 * // return 0 if sum(arr[l..r]) == sum(arr[x..y])
 * // return -1 if sum(arr[l..r]) < sum(arr[x..y])
 * func (this *ArrayReader) compareSub(l, r, x, y int) int {}
 *
 * // Returns the length of the array
 * func (this *ArrayReader) length() int {}
 */

func getIndex(reader *ArrayReader) int {

}
```
