```java
import java.util.Arrays;

public class Equalarray {
    public static boolean CC(int[] arr1, int[] arr2) {
        Arrays.sort(arr1);
        Arrays.sort(arr2);
        boolean flag = true;
        for(int i = 0; i < arr1.length; i++){
            if(arr1[i] != arr2[i]){
                flag = false;
                break;
            }
        }
        return flag;
    }
    Run | Debug
    public static void main(String[] kapilesh) {
        int[] arr1 = new int[] {1,2,5,4,0};
        int[] arr2 = new int[] {2,4,5,0,1};
        System.out.println(CC(arr1, arr2));
    }
}
```

O(N logN)

```java
public class Floor {
    static int FF(int[] arr, int k) {
        Stack<Integer> holder = new Stack<>();
        for(int i = 0; i < arr.length; i++){

            if(arr[i] <= k){
                if(!holder.isEmpty()){
                    if(holder.peek() < arr[i]){
                        holder.push(i);
                    }
                }
                else{
                    holder.push(i);
                }
            }
        }
        if(!holder.isEmpty()){
            return holder.peek();
        }
        return -1;
    }
    Run | Debug
    public static void main(String[] kapilesh) {
        int[] arr = new int[] {1,2,8,10,11,12,19};
        int k = 5;
        System.out.println(FF(arr, k));
    }
}
```

int i - Floor.FF(int[], int)

O(N)

```java
public class Knapsack {
    static int ksolve(int we, int[] w, int[] p, int n){
        if(n == 0 || we == 0){
            return 0;
        }
        if(w[n-1] > we){
            return ksolve(we, w, p, n-1);
        }
        else{
            return Math.max(ksolve(we, w, p, n-1),p[n-1]+ksolve(we - w[n-1], w, p, n-1));
        }
    }
    Run | Debug
    public static void main(String[] kapilesh) {
        int[] p = new int[] {60,100,120};
        int[] w = new int[] {10,20,30};
        int we = 50;
        System.out.println(ksolve(we,w,p,p.length));
    }
}
```

O(N^2)

```java
public class Node {
    static boolean PL(Node head) {
        while (p != null) {
            holder.push(p.data);
            p = p.next;
        }
        while (head != null) {
            if (head.data == holder.peek()) {
                holder.pop();
            } else {
                return false;
            }
            head = head.next;
        }
        return true;
    }

    Run | Debug
    public static void main(String[] kapilesh) {
        Node head = new Node(da:1);
        head.next = new Node(da:2);
        head.next.next = new Node(da:3);
        head.next.next.next = new Node(da:2);
        head.next.next.next.next = new Node(da:1);

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

O(N)

```java
public class Triplesum {
    static List<List<Integer>> TS(int[] nums) {
        List<List<Integer>> ans = new ArrayList<>();
        Arrays.sort(nums);
        for(int i = 0;i < nums.length;i++){
            if(i > 0 && nums[i] == nums[i-1]){
                continue;
            }
            int j = i + 1;
            int k = nums.length - 1;
            while (j < k) {
                int total = nums[i] + nums[j] + nums[k];
                if (total > 0) {
                    k--;
                } else if (total < 0) {
                    j++;
                } else {
                    ans.add(Arrays.asList(nums[i], nums[j], nums[k]));
                    j++;
                    while (nums[j] == nums[j-1] && j < k) {
                        j++;
                    }
                }
            }
        }
        return ans;
    }
    Run | Debug
    public static void main(String[] kapilesh) {
        int[] arr = new int[] {-1,0,1,2,-1,-4};
        System.out.println(TS(arr));
    }
}
```

O(N^2)