题目描述：让我们来模拟一个消息队列的运作，有一个发布者和若干消费者，发布者会在给定的时刻向消息队列发送消息，若此时消息队列有消费者订阅，这个消息会被发送到订阅的消费者中优先级最高（输入中消费者按优先级升序排列）的一个；若此时没有订阅的消费者，该消息被消息队列丢弃。消费者则会在给定的时刻订阅消息队列或取消订阅。

当消息发送和订阅发生在同一时刻时，先处理订阅操作，即同一时刻订阅的消费者成为消息发送的候选。

当消息发送和取消订阅发生在同一时刻时，先处理取消订阅操作，即消息不会被发送到同一时刻取消订阅的消费者。

输入描述：输入为两行。

第一行为2N个正整数，代表发布者发送的N个消息的时刻和内容（为方便解析，消息内容也用正整数表示）。第一个数字是第一个消息的发送时刻，第二个数字是第一个消息的内容，以此类推。用例保证发送时刻不会重复，但注意消息并没有按照发送时刻排列。

第二行为2M个正整数，代表M个消费者订阅和取消订阅的时刻。第一个数字是第一个消费者订阅的时刻，第二个数字是第一个消费者取消订阅的时刻，以此类推。用例保证每个消费者的取消订阅时刻大于订阅时刻，消费者按优先级升序排列。

两行的数字都由空格分隔。N不超过100，M不超过10，每行的长度不超过1000字符。

输出描述：输出为M行，依次为M个消费者收到的消息内容，消息内容按收到的顺序排列，且由空格分隔；若某个消费者没有收到任何消息，则对应的行输出-1。

补充说明：

示例1
输入：2 22 1 11 4 44 5 55 3 33
1 7 2 3
输出：11 33 44 55
22
说明：消息11在1时刻到达，此时只有第一个消费者订阅，消息发送给它；消息22在2时刻到达，此时两个消费者都订阅了，消息发送给优先级最高的第二个消费者；消息33在时刻3到达，此时只有第一个消费者订阅，消息发送给它；余下的消息按规则也是发送给第一个消费者。

示例2
输入：5 64 11 64 9 97
9 11 4 9
输出：97
64
说明：消息64在5时刻到达，此时只有第二个消费者订阅，消息发送给它；消息97在9时刻到达，此时只有第一个消费者订阅（因为第二个消费者刚好在9时刻取消订阅），消息发送给它；11时刻也到达了一个内容为64的消息，不过因为没有消费者订阅，消息被丢弃。

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int[] pub = Arrays.stream(scanner.nextLine().split(" ")).mapToInt(Integer::parseInt).toArray();
        int[] sub = Arrays.stream(scanner.nextLine().split(" ")).mapToInt(Integer::parseInt).toArray();
        getResult(pub, sub);

    }

    private static void getResult(int[] pub, int[] sub) {
        int n = pub.length;
        int m = sub.length;
        int[][] pubArr = new int[n / 2][];
        for (int i = 0, j = 0; i < n; ) {
```

```
            pubArr[j] = new int[]{pub[i], pub[i + 1]};
            j = j + 1;
            i = i + 2;
        }
        int[][] subArr = new int[m / 2][];
        for (int i = 0, j = 0; i < m; ) {

            subArr[j] = new int[]{sub[i], sub[i + 1]};
            j = j + 1;
            i = i + 2;
        }
        Arrays.sort(pubArr, (a, b) -> a[0] - b[0]);
        List<List<Integer>> subList = new ArrayList<>();
        for (int i = 0; i <subArr.length ; i++) {
            subList.add(new ArrayList<>());
        }

        for (int[] pubTemp : pubArr) {
            int t = pubTemp[0];
            int v = pubTemp[1];
            for (int i = subArr.length-1; i >=0 ; i--) {
                int start = subArr[i][0];
                int end = subArr[i][1];
                if(t >= start && t < end){
                    subList.get(i).add(v);
                    break;
                }
            }

        }

        for (List<Integer> integers : subList) {
            if(integers.size() == 0){
                System.out.println("-1");
            } else {
                StringBuilder sb = new StringBuilder();
                for (Integer integer : integers) {
                    sb.append(integer).append(" ");
                }
                System.out.println(sb.toString());
            }
        }
    }
}
```