Java-TLV 解码

题目描述：

TLV 编码是按[Tag Length Value]格式进行编码的，一段码流中的信元用 Tag 标识，Tag 在码流中唯一不重复，Length 表示信元 Value 的长度，Value 表示信元的值。

码流以某信元的 Tag 开头，Tag 固定占一个字节，Length 固定占两个字节，字节序为小端序。

现给定 TLV 格式编码的码流，以及需要解码的信元 Tag，请输出该信元的 Value。

输入码流的 16 机制字符中，不包括小写字母，且要求输出的 16 进制字符串中也不要包含小写字母；码流字符串的最大长度不超过 50000 个字节。

输入描述：

输入的第一行为一个字符串，表示待解码信元的 Tag；
输入的第二行为一个字符串，表示待解码的 16 进制码流，字节之间用空格分隔。

输出描述：

输出一个字符串，表示待解码信元以 16 进制表示的 Value。

补充说明：

示例 1
输入：

```
31
32 01 00 AE 90 02 00 01 02 30 03 00 AB 32 31 31 02 00 32 33 33 01 00 CC
```

输出：

```
32 33
```

说明：

需要解析的信元的 Tag 是 31，从码流的起始处开始匹配，Tag 为 32 的信元长度为 1（01 00，小端序表示为 1）；第二个信元的 Tag 是 90，其长度为 2；第三个信元的 Tag 是 30，其长度为 3；第四个信元的 Tag 是 31，其长度为 2（02 00），所以返回长度后面的两个字节即可，即 32 33。

```
import java.util.Scanner;

// 注意类名必须为 Main, 不要有任何 package xxx 信息
public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        String quest = in.nextLine();
        String stream = in.nextLine();
        String[] streamArr = stream.split(" ");
        int point = 0;
        while (!streamArr[point].equals(quest)) {
            Integer number = Integer.parseInt(streamArr[point + 1], 16) +
Integer.parseInt(streamArr[point + 2], 16) * 256;
```

```
                point = point + 3 + number;
                if (point >= streamArr.length) {
                    break;
                };
            }
            if (point >= streamArr.length) {
                System.out.println("no");
                return;
            }
            Integer number = Integer.parseInt(streamArr[point + 1], 16) +
Integer.parseInt(streamArr[point + 2], 16) * 256;
            for (Integer i = point + 3; i < point + 3 + number; i++) {
                System.out.print(streamArr[i]);
                if (i != point + 2 + number) {
                    System.out.print(" ");
                }
            }
        }
}
```