

We Are A Team 题目描述:

总共有 n 个人在机房, 每个人有一个标号 ($1 \leq \text{标号} \leq n$), 他们分成了多个团队, 需要你根据收到的 m 条消息判定指定的两个人是否在一个团队中, 具体的:

- 1、消息构成为: $a\ b\ c$, 整数 a 、 b 分别代表了两个人的标号, 整数 c 代表指令。
- 2、 $c==0$ 代表 a 和 b 在一个团队内。
- 3、 $c==1$ 代表需要判定 a 和 b 的关系, 如果 a 和 b 是一个团队, 输出一行“*we are a team*”, 如果不是, 输出一行“*we are not a team*”。
- 4、 c 为其它值, 或当前行 a 或 b 超出 $1 \sim n$ 的范围, 输出“*da pian zi*”。

输入描述:

- 1、第一行包含两个整数 n, m ($1 \leq n, m \leq 100000$), 分别表示有 n 个人和 m 条消息。
- 2、随后的 m 行, 每行一条消息, 消息格式为: $a\ b\ c$ ($1 \leq a, b \leq n, 0 \leq c \leq 1$)。

输出描述:

- 1、 $c==1$ 时, 根据 a 和 b 是否在一个团队中输出一行字符串, 在一个团队中输出“*we are a team*”, 不在一个团队中输出“*we are not a team*”。
- 2、 c 为其他值, 或当前行 a 或 b 的标号小于 1 或者大于 n 时, 输出字符串“*da pian zi*”。
- 3、如果第一行 n 和 m 的值超出约定的范围时, 输出字符串“*NULL*”。

补充说明:

示例 1

输入:

```
5 6
1 2 0
1 2 1
1 5 0
```

```
2 3 1
2 5 1
1 3 2
```

输出:

```
we are a team
we are not a team
we are a team
da pian zi
```

说明:

第 2 行定义了 1 和 2 是一个团队

第 3 行要求进行判定, 输出"we are a team"

第 4 行定义了 1 和 5 是一个团队, 自然 2 和 5 也是一个团队

第 5 行要求进行判定, 输出"we are not a team"

第 6 行要求进行判定, 输出"we are a team"

第 7 行 c 为其它值, 输出"da pian zi"

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

```
import java.util.stream.Collectors;
```

```
// 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) {
```

```
        // 注意 hasNext 和 hasNextLine 的区别
```

```
        Scanner scanner = new Scanner(System.in);
```

```
        Map<Integer, Set<Integer>> map = new HashMap<>();
```

```
        List<Integer> collect = Arrays.stream(scanner.nextLine().split("
")).map(Integer::parseInt).collect(Collectors.toList());
```

```
        int n = collect.get(0);
```

```

int m = collect.get(1);

if ((n < 1 || n > 1000000) || (m < 1 || m > 100000)){

    System.out.println("NULL");

}else {

    for (int i = 0; i < m; i++) {

        String message = scanner.nextLine();

        List<Integer>    msg    =    Arrays.stream(message.split("

    ")).map(Integer::parseInt).collect(Collectors.toList());

        int p1 = msg.get(0);

        int p2 = msg.get(1);

        int cmd = msg.get(2);

        String r = fun(p1, p2, cmd, map, n);

        if (r != null){

            System.out.println(r);

        }

    }

}

}

public static String fun(int p1,int p2,int cmd,Map<Integer,Set<Integer>>
map,int n){

    if ((p1 < 1 || p1 > n) || (p2 < 1 || p2 > n)){

        return "da pian zi";
    }
}

```

```

}

if (cmd == 0){

    Set<Integer> set1 = map.get(p1);

    Set<Integer> set2 = map.get(p2);

    if (set1 == null && set2 == null){

        Set<Integer> newSet = new HashSet<>();

        newSet.add(p1);

        newSet.add(p2);

        map.put(p1,newSet);

        map.put(p2,newSet);

    }else if (set1 != null && set2 != null){

        Set<Integer> newSet = new HashSet<>();

        newSet.addAll(set1);

        newSet.addAll(set2);

        map.put(p1,newSet);

        map.put(p2,newSet);

    } else if (set1 != null){

        set1.add(p2);

        map.put(p2,set1);

    }else {

        set2.add(p1);

        map.put(p1,set2);

    }

}

```

```
    }

    }else if (cmd == 1){

        Set<Integer> set1 = map.get(p1);

        Set<Integer> set2 = map.get(p2);

        if (set1 != null && set2 != null && set1.contains(p2)){

            return "we are a team";

        }else {

            return "we are not a team";

        }

    }else {

        return "da pian zi";

    }

    return null;

}

}
```