

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.ArrayList;
import java.util.List;
import java.util.*;
public class Main {
```

```

public static void main(String[] args) {
    Scanner in=new Scanner(System.in);
    int n=in.nextInt();
    int len=in.nextInt();
    if (n<1 || n>100000 || len<1 || len>100000){
        System.out.println("NULL");
        return;
    }
    int[][] messages=new int[len][];
    for (int i = 0; i < len; i++) {
        int a=in.nextInt();
        int b=in.nextInt();
        int c=in.nextInt();
        messages[i]=new int[]{a,b,c};
    }
    Union union = new Union(n + 1);
    List<String> res=new ArrayList<>();
    for (int i = 0; i < messages.length; i++) {
        int[] message = messages[i];
        int a=message[0];
        int b=message[1];
        int c=message[2];
        if ((c!=0 && c!=1) || a>n || b>n || a<1 || b<1){
            res.add("da pian zi");
            continue;
        }
        if (c==0){
            union.union(a,b);
            continue;
        }
        if (c==1){
            if (union.isTeam(a,b)) {
                res.add("we are a team");
            }else{
                res.add("we are not a team");
            }
        }
    }
    StringJoiner sj=new StringJoiner("\n");
    res.stream().forEach(e->sj.add(e));
    System.out.println(sj.toString());
}

```

```

static class Union{
    int[] parent;
    int size;
    Union(int size){
        this.size=size;
        this.parent=new int[size];
        for (int i = 0; i < parent.length; i++) {
            parent[i]=i;
        }
    }

    void union(int a,int b){
        int p1 = findParent(a);
        int p2=findParent(b);
        if (p1==p2)return;
        parent[p1]=p2;
    }

    boolean isTeam(int a,int b){
        int p1 = findParent(a);
        int p2 = findParent(b);
        if (p1==p2)return true;
        return false;
    }

    int findParent(int a){
        if (parent[a]!=a){
            parent[a]=findParent(parent[a]);
        }
        return parent[a];
    }
}

```