**哈密顿绕行世界问题**

**Time Limit: 3000/1000 MS (Java/Others)    Memory Limit: 32768/32768 K (Java/Others)  
Total Submission(s): 6269    Accepted Submission(s): 3760**

**Problem Description**

一个规则的实心十二面体，它的 20个顶点标出世界著名的20个城市，你从一个城市出发经过每个城市刚好一次后回到出发的城市。

**Input**

前20行的第i行有3个数,表示与第i个城市相邻的3个城市.第20行以后每行有1个数m,m<=20,m>=1.m=0退出.

**Output**

输出从第m个城市出发经过每个城市1次又回到m的所有路线,如有多条路线,按字典序输出,每行1条路线.每行首先输出是第几条路线.然后个一个: 后列出经过的城市.参看Sample output

**Sample Input**

2 5 20

1 3 12

2 4 10

3 5 8

1 4 6

5 7 19

6 8 17

4 7 9

8 10 16

3 9 11

10 12 15

2 11 13

12 14 20

13 15 18

11 14 16

9 15 17

7 16 18

14 17 19

6 18 20

1 13 19

5

0

**Sample Output**

1: 5 1 2 3 4 8 7 17 18 14 15 16 9 10 11 12 13 20 19 6 5

2: 5 1 2 3 4 8 9 10 11 12 13 20 19 18 14 15 16 17 7 6 5

3: 5 1 2 3 10 9 16 17 18 14 15 11 12 13 20 19 6 7 8 4 5

4: 5 1 2 3 10 11 12 13 20 19 6 7 17 18 14 15 16 9 8 4 5

5: 5 1 2 12 11 10 3 4 8 9 16 15 14 13 20 19 18 17 7 6 5

6: 5 1 2 12 11 15 14 13 20 19 18 17 16 9 10 3 4 8 7 6 5

7: 5 1 2 12 11 15 16 9 10 3 4 8 7 17 18 14 13 20 19 6 5

8: 5 1 2 12 11 15 16 17 18 14 13 20 19 6 7 8 9 10 3 4 5

9: 5 1 2 12 13 20 19 6 7 8 9 16 17 18 14 15 11 10 3 4 5

10: 5 1 2 12 13 20 19 18 14 15 11 10 3 4 8 9 16 17 7 6 5

11: 5 1 20 13 12 2 3 4 8 7 17 16 9 10 11 15 14 18 19 6 5

12: 5 1 20 13 12 2 3 10 11 15 14 18 19 6 7 17 16 9 8 4 5

13: 5 1 20 13 14 15 11 12 2 3 10 9 16 17 18 19 6 7 8 4 5

14: 5 1 20 13 14 15 16 9 10 11 12 2 3 4 8 7 17 18 19 6 5

15: 5 1 20 13 14 15 16 17 18 19 6 7 8 9 10 11 12 2 3 4 5

16: 5 1 20 13 14 18 19 6 7 17 16 15 11 12 2 3 10 9 8 4 5

17: 5 1 20 19 6 7 8 9 10 11 15 16 17 18 14 13 12 2 3 4 5

18: 5 1 20 19 6 7 17 18 14 13 12 2 3 10 11 15 16 9 8 4 5

19: 5 1 20 19 18 14 13 12 2 3 4 8 9 10 11 15 16 17 7 6 5

20: 5 1 20 19 18 17 16 9 10 11 15 14 13 12 2 3 4 8 7 6 5

21: 5 4 3 2 1 20 13 12 11 10 9 8 7 17 16 15 14 18 19 6 5

22: 5 4 3 2 1 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5

23: 5 4 3 2 12 11 10 9 8 7 6 19 18 17 16 15 14 13 20 1 5

24: 5 4 3 2 12 13 14 18 17 16 15 11 10 9 8 7 6 19 20 1 5

25: 5 4 3 10 9 8 7 6 19 20 13 14 18 17 16 15 11 12 2 1 5

26: 5 4 3 10 9 8 7 17 16 15 11 12 2 1 20 13 14 18 19 6 5

27: 5 4 3 10 11 12 2 1 20 13 14 15 16 9 8 7 17 18 19 6 5

28: 5 4 3 10 11 15 14 13 12 2 1 20 19 18 17 16 9 8 7 6 5

29: 5 4 3 10 11 15 14 18 17 16 9 8 7 6 19 20 13 12 2 1 5

30: 5 4 3 10 11 15 16 9 8 7 17 18 14 13 12 2 1 20 19 6 5

31: 5 4 8 7 6 19 18 17 16 9 10 3 2 12 11 15 14 13 20 1 5

32: 5 4 8 7 6 19 20 13 12 11 15 14 18 17 16 9 10 3 2 1 5

33: 5 4 8 7 17 16 9 10 3 2 1 20 13 12 11 15 14 18 19 6 5

34: 5 4 8 7 17 18 14 13 12 11 15 16 9 10 3 2 1 20 19 6 5

35: 5 4 8 9 10 3 2 1 20 19 18 14 13 12 11 15 16 17 7 6 5

36: 5 4 8 9 10 3 2 12 11 15 16 17 7 6 19 18 14 13 20 1 5

37: 5 4 8 9 16 15 11 10 3 2 12 13 14 18 17 7 6 19 20 1 5

38: 5 4 8 9 16 15 14 13 12 11 10 3 2 1 20 19 18 17 7 6 5

39: 5 4 8 9 16 15 14 18 17 7 6 19 20 13 12 11 10 3 2 1 5

40: 5 4 8 9 16 17 7 6 19 18 14 15 11 10 3 2 12 13 20 1 5

41: 5 6 7 8 4 3 2 12 13 14 15 11 10 9 16 17 18 19 20 1 5

42: 5 6 7 8 4 3 10 9 16 17 18 19 20 13 14 15 11 12 2 1 5

43: 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 1 2 3 4 5

44: 5 6 7 8 9 16 17 18 19 20 1 2 12 13 14 15 11 10 3 4 5

45: 5 6 7 17 16 9 8 4 3 10 11 15 14 18 19 20 13 12 2 1 5

46: 5 6 7 17 16 15 11 10 9 8 4 3 2 12 13 14 18 19 20 1 5

47: 5 6 7 17 16 15 11 12 13 14 18 19 20 1 2 3 10 9 8 4 5

48: 5 6 7 17 16 15 14 18 19 20 13 12 11 10 9 8 4 3 2 1 5

49: 5 6 7 17 18 19 20 1 2 3 10 11 12 13 14 15 16 9 8 4 5

50: 5 6 7 17 18 19 20 13 14 15 16 9 8 4 3 10 11 12 2 1 5

51: 5 6 19 18 14 13 20 1 2 12 11 15 16 17 7 8 9 10 3 4 5

52: 5 6 19 18 14 15 11 10 9 16 17 7 8 4 3 2 12 13 20 1 5

53: 5 6 19 18 14 15 11 12 13 20 1 2 3 10 9 16 17 7 8 4 5

54: 5 6 19 18 14 15 16 17 7 8 9 10 11 12 13 20 1 2 3 4 5

55: 5 6 19 18 17 7 8 4 3 2 12 11 10 9 16 15 14 13 20 1 5

56: 5 6 19 18 17 7 8 9 16 15 14 13 20 1 2 12 11 10 3 4 5

57: 5 6 19 20 1 2 3 10 9 16 15 11 12 13 14 18 17 7 8 4 5

58: 5 6 19 20 1 2 12 13 14 18 17 7 8 9 16 15 11 10 3 4 5

59: 5 6 19 20 13 12 11 10 9 16 15 14 18 17 7 8 4 3 2 1 5

60: 5 6 19 20 13 14 18 17 7 8 4 3 10 9 16 15 11 12 2 1 5

**import** java.util.\*;

**public** **class** Main

{

**static** **boolean**[][] *key*=**new** **boolean**[25][25];

**static** **boolean**[] *key1*=**new** **boolean**[25];

**static** **int**[] *next*=**new** **int**[25];

**static** **int** *count*=1;

**static** **int** *count1*=1;

**static** **int** *start*=1;

**static** **int** *sum*=0;

**static** Scanner *in*=**new** Scanner(System.***in***);

**public** **static** **void** main (String[] args)

{

**for**(**int** i=1;i<=20;i++)

{

**for**(**int** j=0;j<3;j++)

{

**int** n=*in*.nextInt();

*key*[i][n]=**true**;

}

}//标记从i点出发可以到达的地点n

**while**(*start*!=0)

{

*count*=1;

*sum*=0;//数据初始化 count为一次搜索中已到达点个数 sum为当前case的序号

*start*=*in*.nextInt();

**for**(**int** i=0;i<25;i++)

{

*key1*[i]=**false**;

}

*key1*[*start*]=**true**;//为起始点设置为已经拜访过

*find*(*start*);

}//读取起始点当输入为0时结束

}

**static** **void** find(**int** a)

{

**if**(*count*==20&&*key*[a][*start*])//

{

*sum*++;

System.***out***.print(*sum*+": ");

*print*(*start*);

}

**else**

**for**(**int** i=1;i<=20;i++)//从1开始遍历所有点，若可以从a点到达i点，则

{

**if**(*key*[a][i]&&!*key1*[i])

{

*key1*[i]=**true**;

*next*[a]=i;

*count*++;

//System.out.println(i+" "+count);

*find*(i);

*count*--;

*key1*[i]=**false**;//一切回归原状态

}

}

}

**static** **void** print(**int** a)

{

System.***out***.print(a+" ");

*count1*++;

**if**(*count1*<=20)

*print*(*next*[a]);

**else** {

*count1*=1;

System.***out***.println(*start*);

}

}

}