Marcos decidiu abandonar o bar da faculdade onde jogava truco para dedicar-se ao mundo da programação. Para que isso fosse mais fácil, decidiu criar um novo jogo de cartas.

O coração das cartas, como Marcos apelidou o jogo, é individual e jogado com três pilhas, inicialmente com o mesmo número de cartas. Cada carta tem um valor numérico inteiro de 0 até 9. O jogador pode, a qualquer momento ver o valor de qualquer carta, mas só pode jogar com as cartas que estão no topo das pilhas. Em cada rodada, o jogador pode remover qualquer combinação de cartas que estejam no topo da pilha (pode escolher 1, 2 ou até 3 cartas) cuja soma dos valores seja múltiplo de 3. O jogo é ganho quando todas as cartas forem removidas das pilhas. Se alguma carta não puder ser removida, perde-se o jogo.

**Entrada**

A entrada é composta por várias instâncias Cada instância é iniciada por um inteiro N (0 ≤ N ≤ 100), que identifica o número de cartas em cada pilha. A entrada termina quando N = 0. Cada uma das N linhas seguintes contém três inteiros A, B e C, que descrevem os valores numéricos das cartas em um nível da pilha (0 ≤ A, B, C ≤  9). As pilhas são descritas do topo até o fundo.

**Saída**

Para cada instância, imprima uma linha contendo o número 1 se o jogador pode ganhar a instância do jogo ou o número 0 se o jogo for impossível.

| Exemplo de Entrada | Exemplo de Saída |
| --- | --- |
| 2 1 1 1 2 0 4 3 1 0 0 0 1 0 0 0 0 0 | 1 0 |

***Solução***

import java.util.\*;

public class Main {

static Map<String, Integer> m = new HashMap<String, Integer>();

static int n;

static int[][] pilha = new int[3][102];

static public boolean cartas(int a, int b, int c) {

String ss = "" + a + b + c;

if(a == b && b == c && c == n) {

m.putIfAbsent(ss, 1);

return true;

}

int x = m.getOrDefault(ss, 0);

if (x > 0) return (x == 1? true : false);

if(a < n && pilha[0][a]%3 == 0 && cartas(a + 1, b, c)) {

m.putIfAbsent(ss, 1);

return true;

}

if(b < n && pilha[1][b]%3 == 0 && cartas(a, b + 1, c)) {

m.putIfAbsent(ss, 1);

return true;

}

if(c < n && pilha[2][c]%3 == 0 && cartas(a, b, c + 1)) {

m.putIfAbsent(ss, 1);

return true;

}

if(a < n && b < n && (pilha[0][a]+pilha[1][b]) % 3 == 0 && cartas(a + 1, b + 1, c)) {

m.putIfAbsent(ss, 1);

return true;

}

if(a < n && c < n && (pilha[0][a]+pilha[2][c]) % 3 == 0 && cartas(a + 1, b, c + 1)) {

m.putIfAbsent(ss, 1);

return true;

}

if(b < n && c < n && (pilha[1][b]+pilha[2][c])%3 == 0 && cartas(a, b + 1, c + 1)) {

m.putIfAbsent(ss, 1);

return true;

}

if(a < n && b < n && c < n && (pilha[0][a]+pilha[1][b]+pilha[2][c]) % 3 == 0 && cartas(a + 1, b + 1, c + 1)) {

m.putIfAbsent(ss, 1);

return true;

}

m.putIfAbsent(ss, 2);

return false;

}

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

while(true) {

n = Integer.parseInt(in.nextLine());

if (n == 0) break;

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

StringTokenizer st = new StringTokenizer(in.nextLine());

pilha[0][i] = Integer.parseInt(st.nextToken());

pilha[1][i] = Integer.parseInt(st.nextToken());

pilha[2][i] = Integer.parseInt(st.nextToken());

}

if (cartas(0,0,0) == true) System.out.println("1");

else System.out.println("0");

m.clear();

}

}

}