O senhor Milli, morador da cidade Petland, é o famoso proprietário da maior fábrica de jogos de tabuleiros do mundo.

Recentemente, ele teve a ideia de lançar um novo jogo exclusivo de tabuleiro, que ele apelidou de Tabuleiro da Frequência.

O jogo ocorre da seguinte forma. Inicialmente, um tabuleiro com dimensões N × N é dado contendo apenas 0’s. Depois disso, Q operações são propostas, podendo ser de 4 tipos:

* 1 X R: Atribuir o valor R a todos os números da linha X;
* 2 X R: Atribuir o valor R a todos os números da coluna X;
* 3 X: Imprimir o valor mais frequente na linha X;
* 4 X: Imprimir o valor mais frequente da coluna X.

Milli não é muito bom com computadores, mas é bastante preguiçoso. Sabendo que você é um dos melhores programadores do mundo, ele precisa sua ajuda para resolver este problema.

**Entrada**

A primeira linha da entrada é composta por dois inteiros N e Q (1 ≤ N, Q ≤ 105), representando, respectivamente, o tamanho do tabuleiro e a quantidade de operações. As próximas Q linhas da entrada vão conter as Q operações. O primeiro inteiro de cada linha vai indicar o tipo da operação. Caso seja 1 ou 2, será seguido por mais dois inteiros X (1 ≤ X ≤ N) e R (0 ≤ R ≤ 50). Caso seja 3 ou 4, será seguido por apenas mais um inteiro X.

**Saída**

Para cada operação do tipo 3 ou 4, seu programa deve produzir uma linha, contendo o valor da resposta correspondente. Se uma linha ou coluna tiver dois ou mais valores que se repetem o mesmo número de vezes, você deve imprimir o maior deles. Por exemplo, se uma linha tem os valores [5,7,7,2,5,2,1,3], tanto o 2, 5 e 7 se repetem duas vezes, então a resposta será 7, pois é o maior deles.

| Exemplo de Entrada | Exemplo de Saída |
| --- | --- |

|  |  |
| --- | --- |
| 2 4  1 1 1  2 2 2  3 1  3 2 | 2  2 |

|  |  |
| --- | --- |
| 3 6  1 1 2  1 2 3  1 3 4  4 3  1 3 0  4 3 | 4  3 |

Código

import java.util.\*;

public class Main {

public static void main(String[] args) {

List<String> entradas = new ArrayList<String>();

Scanner sc = new Scanner(System.in);

while(sc.hasNext()) {

entradas.add(sc.nextLine());

}

Iterator<String> it = entradas.iterator();

String linha = it.next();

Integer tamTab = Integer.parseInt(linha.split(" ")[0]);

Integer qtOp = Integer.parseInt(linha.split(" ")[1]);

List<Map<String, Integer>> operacoes = lerOperacoes(qtOp, it);

Integer tabuleiro[][] = criarTabuleiro(tamTab);

for(Map<String, Integer> operacao : operacoes) {

if(operacao.get("tipoOp") == 1) {

tabuleiro = realizarOperacao1(operacao, tabuleiro, tamTab);

} else if(operacao.get("tipoOp") == 2) {

tabuleiro = realizarOperacao2(operacao, tabuleiro, tamTab);

} else if(operacao.get("tipoOp") == 3) {

Map<Integer, Integer> repeticoesNumero = lerRepeticoesLinhaX(operacao, tabuleiro, tamTab);

Integer maiorNumeroComMaiorRepeticao = avaliaMaiorNumeroComMaiorRepeticao(repeticoesNumero);

System.out.println(maiorNumeroComMaiorRepeticao);

} else if(operacao.get("tipoOp") == 4) {

Map<Integer, Integer> repeticoesNumero = lerRepeticoesColunaX(operacao, tabuleiro, tamTab);

Integer maiorNumeroComMaiorRepeticao = avaliaMaiorNumeroComMaiorRepeticao(repeticoesNumero);

System.out.println(maiorNumeroComMaiorRepeticao);

}

}

}

private static List<Map<String, Integer>> lerOperacoes(Integer qtOp, Iterator<String> it) {

List<Map<String, Integer>> operacoes = new ArrayList<Map<String, Integer>>();

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

String strOperacao = it.next();

Integer tipoOp = Integer.parseInt(strOperacao.split(" ")[0]);

if(tipoOp == 1 || tipoOp == 2) {

Integer numLinColX = Integer.parseInt(strOperacao.split(" ")[1]);

Integer valorR = Integer.parseInt(strOperacao.split(" ")[2]);

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

operacao.put("tipoOp", tipoOp);

operacao.put("numLinColX", numLinColX);

operacao.put("valorR", valorR);

operacoes.add(operacao);

} else if(tipoOp == 3 || tipoOp == 4) {

Integer numLinColX = Integer.parseInt(strOperacao.split(" ")[1]);

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

operacao.put("tipoOp", tipoOp);

operacao.put("numLinColX", numLinColX);

operacoes.add(operacao);

}

}

return operacoes;

}

private static Integer[][] criarTabuleiro(int tamTab) {

Integer tabuleiro[][] = new Integer[tamTab][tamTab];

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

for(int j = 0; j < tamTab; j++) {

tabuleiro[i][j] = 0;

}

}

return tabuleiro;

}

private static void imprimeTabuleiro(Integer[][] tabuleiro, int tamTab) {

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

for(int j = 0; j < tamTab; j++) {

System.out.print(tabuleiro[i][j] + " ");

}

System.out.println();

}

}

private static Integer[][] realizarOperacao1(Map<String, Integer> operacao,

Integer[][] tabuleiro,

int tamTab) {

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

int linhaX = operacao.get("numLinColX") - 1;

tabuleiro[linhaX][i] = operacao.get("valorR");

}

return tabuleiro;

}

private static Integer[][] realizarOperacao2(Map<String, Integer> operacao,

Integer[][] tabuleiro,

int tamTab) {

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

int colunaX = operacao.get("numLinColX") - 1;

tabuleiro[i][colunaX] = operacao.get("valorR");

}

return tabuleiro;

}

private static Map<Integer, Integer> lerRepeticoesLinhaX(Map<String, Integer> operacao,

Integer[][] tabuleiro,

Integer tamTab) {

Map<Integer, Integer> repeticoesNumero = new HashMap<Integer, Integer>();

int linhaX = operacao.get("numLinColX") - 1;

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

Integer pivo = tabuleiro[linhaX][i];

if(repeticoesNumero.containsKey(pivo)) {

Integer qt = repeticoesNumero.get(pivo);

qt++;

repeticoesNumero.put(pivo, qt);

} else {

repeticoesNumero.put(pivo, 1);

}

}

return repeticoesNumero;

}

private static Map<Integer, Integer> lerRepeticoesColunaX(Map<String, Integer> operacao,

Integer[][] tabuleiro,

Integer tamTab) {

Map<Integer, Integer> repeticoesNumero = new HashMap<Integer, Integer>();

int colunaX = operacao.get("numLinColX") - 1;

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

Integer pivo = tabuleiro[i][colunaX];

if(repeticoesNumero.containsKey(pivo)) {

Integer qt = repeticoesNumero.get(pivo);

qt++;

repeticoesNumero.put(pivo, qt);

} else {

repeticoesNumero.put(pivo, 1);

}

}

return repeticoesNumero;

}

public static Integer avaliaMaiorNumeroComMaiorRepeticao(Map<Integer, Integer> repeticoesNumero) {

List<Integer> maioresRepeticoes = new ArrayList<Integer>();

Integer maiorRepeticao = Collections.max (repeticoesNumero.values());

List<Integer> chaves = new ArrayList<Integer>(repeticoesNumero.keySet());

for(Integer chave : chaves) {

Integer valorRepeticao = repeticoesNumero.get(chave);

if(valorRepeticao >= maiorRepeticao) {

maioresRepeticoes.add(chave);

}

}

Integer maiorNumeroComMaiorRepeticao = Collections.max(maioresRepeticoes);

return maiorNumeroComMaiorRepeticao;

}

}