* Create a list, , of  empty sequences, where each sequence is indexed from  to . The elements within each of the  sequences also use -indexing.
* Create an integer, , and initialize it to .
* The  types of queries that can be performed on your list of sequences () are described below:
  1. Query: 1 x y
     1. Find the sequence, , at index  in .
     2. Append integer  to sequence .
  2. Query: 2 x y
     1. Find the sequence, , at index  in .
     2. Find the value of element  in  (where  is the size of ) and assign it to .
     3. Print the new value of  on a new line

**Task**   
Given , , and  queries, execute each query.

**Note:**  is the *bitwise XOR* operation, which corresponds to the ^ operator in most languages. Learn more about it on [Wikipedia](https://en.wikipedia.org/wiki/Exclusive_or).

**Input Format**

The first line contains two space-separated integers,  (the number of sequences) and  (the number of queries), respectively.   
Each of the  subsequent lines contains a query in the format defined above.

**Constraints**

* It is guaranteed that query type  will never query an empty sequence or index.

**Output Format**

For each type  query, print the updated value of  on a new line.

**Sample Input**

2 5

1 0 5

1 1 7

1 0 3

2 1 0

2 1 1

**Sample Output**

7

3

**Explanation**

*Initial Values:*   
   
   
 = [ ]   
 = [ ]

*Query 0:* Append  to sequence .   
   
 = [5]   
 = [ ]

*Query 1:* Append  to sequence .   
 = [5]   
 = [7]

*Query 2:* Append  to sequence .   
   
 = [5, 3]   
 = [7]

*Query 3:* Assign the value at index  of sequence  to , print .   
   
 = [5, 3]   
 = [7]

7

*Query 4:* Assign the value at index  of sequence  to , print .   
   
 = [5, 3]   
 = [7]

3

import java.io.\*;

import java.math.\*;

import java.security.\*;

import java.text.\*;

import java.util.\*;

import java.util.concurrent.\*;

import java.util.regex.\*;

public class Solution {

// Complete the dynamicArray function below.

static List<Integer> dynamicArray(int n, List<List<Integer>> queries) {

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

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

int lastAnswer = 0;

// Initalizing "seqList" with empty Lists

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

seqList.add(new ArrayList<Integer>());

for(int i=0; i<queries.size(); i++) {

if(queries.get(i).get(0) == 1) {

int index = (queries.get(i).get(1) ^ lastAnswer) % n;

List<Integer> seq = seqList.get(index);

seq.add(queries.get(i).get(2));

} else if(queries.get(i).get(0) == 2) {

int index = (queries.get(i).get(1) ^ lastAnswer) % n;

List<Integer> seq = seqList.get(index);

int size = seq.size();

int value = seq.get(queries.get(i).get(2) % size);

lastAnswer = value;

result.add(lastAnswer);

}

}

return result;

}

public static void main(String[] args) throws IOException {

BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));

BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(System.getenv("OUTPUT\_PATH")));

String[] nq = bufferedReader.readLine().replaceAll("\\s+$", "").split(" ");

int n = Integer.parseInt(nq[0]);

int q = Integer.parseInt(nq[1]);

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

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

String[] queriesRowTempItems = bufferedReader.readLine().replaceAll("\\s+$", "").split(" ");

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

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

int queriesItem = Integer.parseInt(queriesRowTempItems[j]);

queriesRowItems.add(queriesItem);

}

queries.add(queriesRowItems);

}

List<Integer> result = dynamicArray(n, queries);

for (int i = 0; i < result.size(); i++) {

bufferedWriter.write(String.valueOf(result.get(i)));

if (i != result.size() - 1) {

bufferedWriter.write("\n");

}

}

bufferedWriter.newLine();

bufferedReader.close();

bufferedWriter.close();

}

}