'''

Arjun wants to build some homes in a land of size R\*C.

He wanted to construct homes in rectangular shape.

The place which is remained will be used for gradening.

Accordingly he has prepared the plan and given as

an 2d array plan[][], where 1 indicates home, and 0 indicates garden area.

A home is set of cells with value 1 in rectangular shape.

He wants to findout all the homes in the plan and store their co-ordinates in

the following order, coords[i] = [x1,y1,x2,y2], where (x1,y1) is the starting

co-ordinate (top left corner), and (x2,y2) is the ending co-ordinate

(bottom right corner) of i-th home.

Your task is to help Arjun to find all the homes and return the coords[][] of

all the homes from top left corner to bottom right corner.

NOTE: No two homes are adjacent to each other in 4 directions,

(left, right, top, bottom).

Input Format:

-------------

Line-1: Two integers R and C, size of the land.

Next R lines: C space separated integers, either 0 or 1

0- represents garden area land and 1- represents the home.

Output Format:

--------------

Print 2d array, the co-ordinates of all homes.

Sample Input-1:

---------------

2 3

1 0 0

0 1 1

Sample Output-1:

----------------

[0, 0, 0, 0][1, 1, 1, 2]

Sample Input-2:

---------------

4 4

1 1 0 1

0 0 0 0

1 1 0 1

1 1 0 1

Sample Output-2:

----------------

[0, 0, 0, 1][0, 3, 0, 3][2, 0, 3, 1][2, 3, 3, 3]

'''

r,c=list(map(int,input().split()))

l=[]

l1=[]

for i in range(0,r):

l1=list(map(int,input().split()))

l.append(l1)

def fun(l,r1,c1):

q=[]

l[r1][c1]=0

rr1=0

cc1=0

q.append([r1,c1])

while(len(q)!=0):

rr1=q[0][0]

cc1=q[0][1]

q.pop(0)

if(cc1+1<c and l[rr1][cc1+1]==1):

l[rr1][cc1+1]=0

q.append([rr1,cc1+1])

if(rr1+1<r and l[rr1+1][cc1]==1 ):

l[rr1+1][cc1]=0

q.append([rr1+1,cc1])

return [rr1,cc1]

sol=[]

for i in range(0,r):

for j in range(0,c):

if(l[i][j]==1):

ans=[]

ans.append(i)

ans.append(j)

ans.extend(fun(l,i,j))

sol.append(ans)

for i in sol:

print(i,end="")

Ibrahim is an interior designer wants to color wall of size M\*N.

He plans to color the wall and put lights of two different colors

The designer can lit the wall using M\*N lights.The lights are Blue or pink

in color.Blue colored lights represented with digit-1 and pink colored lights

represented with digit-0.

The Blue colored lights forms different shapes, that are connected 4 directonally.

The directons are upwards, downwards, left, and right. Ibrahim got an idea to

count the unique shapes formed by blue colored lights.

You will be given the decorated wall as a matrix wall[][].

Your task is to help Ibrahim to count the unique shapes by the lights.

Input Format:

-------------

Line-1: Two space separated integers M and N, size of the wall.

Next M lines: N space separated integers, either 0 or 1.

Output Format:

--------------

Print an integer, Number of distinct shapes formed by Blue Lights.

Sample Input-1:

---------------

4 5

1 1 0 1 1

1 1 0 0 1

0 0 0 0 0

1 1 0 0 0

Sample Output-1:

----------------

3

Sample Input-2:

---------------

5 5

1 1 0 1 1

1 0 0 0 1

0 0 0 0 0

1 0 0 0 1

1 1 0 1 1

Sample Output-2:

----------------

4

Note:

-------

The shapes,

1 1 1 1

1 and 1

import java.util.\*;

public class Main{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

Set<String> hs=new HashSet<>();

int r=sc.nextInt();

int c=sc.nextInt();

int[][] arr=new int[r][c];

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

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

arr[i][j]=sc.nextInt();

}

}

int count=0;

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

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

if(arr[i][j]==1){

StringBuilder s=new StringBuilder("");

dfs(arr,i,j,s,i,j);

// count+=1;

hs.add(s.toString());

// System.out.println(s);

}

}

}

System.out.println(hs.size());

}

public static void dfs(int[][] arr,int r,int c,StringBuilder s,int rr,int cc){

arr[r][c]=0;

s.append(String.valueOf(r-rr));

s.append(String.valueOf(c-cc));

if(r-1>=0 && arr[r-1][c]==1){

dfs(arr,r-1,c,s,rr,cc);

}

if(c+1<arr[0].length && arr[r][c+1]==1 ){

// arr[r][c+1]=0;

dfs(arr,r,c+1,s,rr,cc);

}

if(c-1>=0 && arr[r][c-1]==1){

// arr[r][c-1]=0

dfs(arr,r,c-1,s,rr,cc);

}

if(r+1<arr.length && arr[r+1][c]==1){

dfs(arr,r+1,c,s,rr,cc);

}

}

}

Vihaar is working with strings.

He is given two strings A and B, and another string T,

where the length of A and B is same.

You can find the relative groups of letters from A and B,

using the following rule set:

- Equality rule: 'p' == 'p'

- Symmetric rule: 'p' == 'q' is same as 'q' == 'p'

- Transitive rule: 'p' == 'q' and 'q' == 'r' indicates 'p' == 'r'.

Vihaar has to form the relatively smallest string of T,

using the relative groups of letters.

For example, if A ="pqr" and B = "rst" ,

then we have 'p' == 'r', 'q' == 's', 'r' == 't' .

The relatives groups formed using above rule set are as follows:

[p, r, t] and [q,s] and String T ="tts", then relatively smallest string is "ppq".

You will be given the strings A , B and T.

Your task is to help Vihaar to find the relatively smallest string of T.

Input Format:

-------------

Three space separated strings, A , B and T

Output Format:

--------------

Print a string, relatively smallest string of T.

Sample Input-1:

---------------

kmit ngit mgit

Sample Output-1:

----------------

ggit

Explanation:

------------

The relative groups using A nd B are [k, n], [m, g], [i], [t] and

the relatively smallest string of T is "ggit"

Sample Input-2:

---------------

attitude progress apriori

Sample Output-2:

----------------

aaogoog

Explanation:

------------

The relative groups using A nd B are [a, p], [t, r, o], [i, g] and [u, e, d, s]

the relatively smallest string of T is "aaogoog"

// this code must be done using dsu so only 62.5 test cases have passed

import java.util.\*;

public class Main{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

String s1=sc.next();

String s2=sc.next();

String s3=sc.next();

// Queue<Character> q=new PriorityQueue<>();

ArrayList<Queue<Character>> l=new ArrayList<>();

for(int i=0;i<s1.length();i++){

int count=0;

for (int j=0;j<l.size();j++){

if(l.get(j).contains(s1.charAt(i))|| l.get(j).contains(s2.charAt(i))){

l.get(j).add(s1.charAt(i));

l.get(j).add(s2.charAt(i));

count+=1;

break;

}

}

if(count==0){

Queue<Character> al=new PriorityQueue<>();

al.add(s1.charAt(i));

al.add(s2.charAt(i));

l.add(al);

}

}

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

// for(int j=i+1;j<l.size();j++){

// if(fun(l.get(i),l.get(j))){

// for(int k=0;k<l.get(j).size();k++){

// l.get(i).add(l.get(j).get(k));

// }

// l.get(j).clear();

// }

// }

// }

// System.out.println(l);

StringBuilder ss=new StringBuilder("");

for (int i=0;i<s3.length();i++){

for(int j=0;j<l.size();j++){

if(l.get(j).contains(s3.charAt(i))){

ss.append(l.get(j).peek());

break;

}

}

}

System.out.println(ss);

}

// public static boolean fun(Queue<Character> q1,Queue<Character> q2){

// for(int i=0;i<q2.size();i++){

// if(q1.contains(q2.get(i))){

// return true;

// }

// }

// return false;

// }

}

Actual solution for the above question

import java.util.\*;

public class Main{

public static int[] parent;

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

String s1=sc.next();

parent=new int[26];

String s2=sc.next();

String s3=sc.next();

for(int i=0;i<26;i++){

parent[i]=i;

}

for(int i=0;i<s1.length();i++){

union(s1.charAt(i)-'a',s2.charAt(i)-'a');

}

StringBuilder sb=new StringBuilder("");

for(int i=0;i<s3.length();i++){

sb.append((char)('a'+find(s3.charAt(i)-'a')));

}

System.out.println(sb);

}

public static void union(int i,int j){

int p1=find(i);

int p2=find(j);

if(p1>p2){

parent[p1]=p2;

}

else{

parent[p2]=p1;

}

}

public static int find(int i){

if(parent[i]== i){

return i;

}

return parent[i]=find(parent[i]);

}

}

A game-zone is represented as an infinite 2D grid. At every integer coordinate

(x,y) there is a bag, and the bag contains |x| + |y| number of chocolates in it.

You want grab N chocolates for your kid. You have to select some square area in

the game-zone such a way that the selected area is an axis-aligned square plot

that is centered at (0, 0).

Your task is to find the minimum perimeter of the square area such that

at least it conains N chocolates are inside or on the perimeter of that area.

NOTE:

The value of |x| is defined as: x if x >= 0 -x if x < 0

Constarint:

-----------

1 <= N <= 10^14

Input Format:

-------------

An integer N, number of chocolates.

Output Format:

--------------

Print an integer result.

Sample Input-1:

---------------

5

Sample Output-1:

----------------

8

Explanation:

------------

Sample Input-2:

---------------

56

Sample Output-2:

----------------

16

Explanation:

------------

Sample Input-3:

---------------

62

Sample Output-3:

----------------

24

import java.util.\*;

public class Main{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

long n=sc.nextLong();

System.out.println(binary\_search(n));

}

public static long calc(long r){

return (4\*r\*r\*r)+(6\*r\*r)+2\*r;

}

public static long binary\_search(long n){

long l=0;

long u=(long)Math.sqrt(n)+1;

while(l<u){

long mid=(long)(l+(u-l)/2);

long ans=calc(mid);

if(ans<n){

l=mid+1;

}

else{

u=mid;

}

}

return l\*8;

}

}