Черепашка

На квадратной доске расставлены целые неотрицательные числа. Черепашка, находящаяся в левом верхнем углу, мечтает попасть в правый нижний. При этом она может переползать только в клетку справа или снизу и хочет, чтобы сумма всех чисел, оказавшихся у нее на пути, была бы максимальной. Требуется составить программу, вычисляющую максимальную сумму и ее путь, составленный из букв x (черепашка переходит в правый квадратик) и y (черепашка переходит в нижний квадратик.).

Формат входных данных

Первая строка — M *N* — размер доски.

Далее следует M строк, каждая из которых содержит *N* целых чисел, представляющие доску.

Формат выходных данных: Одно число — максимальная сумма.

Пример

INPUT.TXT	OUTPUT.TXT
3 3 1 2 3 1 2 3 1 2 3	12 Xxyy

Решение: для определения максимальной суммы определим в программе целочисленную таблицу z, а для формирования оптимального пути черепашки – символьную таблицу u, в каждую клетку которой (кроме клетки (1, 1)) – один из символов x или y, определяющий направление движения черепашки из предыдущей клетки оптимального пути. Размеры этих таблиц совпадают с размерами таблицы a.

A

1	2	3
1	2	3
1	2	3

Z

U

Очевидно z[1, 1] = a[1, 1], значение элемента u[1, 1] таблицы u может быть любым.

Оптимальные маршруты из клетки (1, 1) в клетки (1, 2) и (2, 1) определяются однозначно:

z[1, 2] = z[1, 1] + a[1, 2], u[1, 2] = ‘x’,

z[2, 1] = z[1, 1] + a[2, 1], u[2, 1] = ‘y’.

Аналогично определяются значения элементов z[1,j], u[1,j], z[i, 1], u[i, 1]:

z[1, j] = z[1, j-1] + a[1, j], u[1, j] = ‘x’,

z[i, 1] = z[i-1, 1] + a[i, 1], u[i, 1] = ‘y’.

A

1	2	3
1	2	3
1	2	3

Z

1	3	6
2		
3		

U

	X	x
Y		
Y		

Определим рекуррентные соотношения, позволяющие вычислять значения элементов таблиц z и u. Рассмотрим произвольную клетку (i, j) заданного

прямоугольного участка, если i > 1 и j > 1. В нее черепашка может придти или из клетки (i-1, j) или из клетки (i, j-1).

	(i-1,j)	
(I,j-1)	**(I,j)**	

Если z[i-1,j] > z[i, j-1], то черепашке лучше двигаться вниз и поэтому z[i, j] = z[i-1, j] + a[i, j], u[i, j] = ‘y’. В противном случае z[i, j] = z[i, j-1] + a[i, j], u[i, j] = ‘x’.

A

1	2	3
1	2	3
1	2	3

Z

1	3	6
2	5	9
3	7	12

U

	x	x
Y	y	y
y	y	y

Восстановим оптимальный путь черепашки “с конца” с помощью таблицы u. Склеивая символы ‘x’ или ’y’ из соответствующих клеток этой таблицы, мы сформируем символьную строку s, содержащую m + n - 2 символов - оптимальный путь черепашки.

(В приведенной программе считывается массив a, затем пересчитывается по реккурентной формуле)

```
program turtle; uses crt;
var n,m,k,i,j:integer; s:string;
   a:array[1..20,1..20] of integer;   u:array[1..20,1..20] of   char;
 begin
  assign(input,'input.txt');
 reset(input); readln(m,n);
for i:=1 to m do for j:=1 to n do  read(a[i,j]);  close(input);
 {------------1 stroka+1stolbez-------------}
for i:=2 to m do begin a[i,1]:=a[i,1]+a[i-1,1]; u[i,1]:='y';end;
for j:=2 to n do begin a[1,j]:=a[1,j]+a[1,j-1]; u[1,j]:='x'; end;
{--------zapolnenie ost_chasti-------------}
for i:=2 to m do for j:=2 to n do
if a[i,j-1]>a[i-1,j] then begin a[i,j]:=a[i,j]+a[i,j-1]; u[i,j]:='x'end
else  begin a[i,j]:=a[i,j]+a[i-1,j]; u[i,j]:='y'end;
{----------put'---------------------------}
s:='';   i:=m; j:=n;
for k:= m+n-2 downto 1 do
begin s:=u[i,j]+s;
if u[i,j]='y' then dec(i) else dec(j);
end;
assign(output,'output.txt'); rewrite(output); writeln( a[m,n]);
writeln( s); close(output);
end.
```