



def getNext(pattern):*#获取模式串pattern改进的next数组* j = 0 *#索引值* k = -1 *#最长相同的前后缀子串长度* next = [len(pattern)]  
 next[0] = -1  
 while j < len(pattern)-1:  
 if k == -1 or pattern[j] == pattern[k]:  
 j += 1  
 k += 1  
 if pattern[j] == pattern[k]:*#pk与pj进行比较* next.insert(j,next[k])  
 else:  
 next.insert(j,k)  
 else:  
 k = next[k]  
 return next  
def indexOf(target, pattern, begin):*#返回target目标串中首个与pattern模式串匹配的子串序号，匹配失败返回-1* n = len(target)  
 m = len(pattern)  
 count = 0  
 if begin < 0:  
 begin = 0  
 if n < 0 or n < m or begin >= n:*#目标串为空、较短或begin越界，则不比较* return -1  
 next = getNext(pattern)  
 i = begin  
 j =0 *# i为target比较字符的下标，j为pattern比较字符的下标* while i<n and j<m:  
 if j == -1 or target[i] == pattern[j]:*#字符相等，则继续比较后续字符* i += 1  
 j += 1  
 else: *#下次匹配，i不回溯* j = next[j]  
 if n-i+1 < m-i+1: *#目标串剩余子串长度不够，不再比较* break  
 if j == m:  
 count += 1  
 j = 0  
 if count > 0:  
 return count  
 else:  
 return 0  
def do(pattern, n):  
 target = []  
 out = []  
 for i in range(n):  
 target.append(input())  
 for j in range(n):  
 out.append(indexOf(target[j], pattern, 0))  
 for m in range(n):  
 print(out[m])  
  
get\_in = input().split(**" "**,1)  
pattern = get\_in[0]  
n = int(get\_in[1])  
do(pattern, n)

**import** java.util.ArrayDeque;  
**import** java.util.Queue;  
*//1 1 3 1  
//2***import** java.util.Scanner;  
  
**public class** HorseCount {  
 **public static void** main(String[] args) {  
 Scanner sc=**new** Scanner(System.***in***);  
 String[] size=sc.nextLine().split(**" "**);  
 **int** sizex=Integer.*parseInt*(size[0]);  
 **int** sizey=Integer.*parseInt*(size[1]);  
 String[] str=sc.nextLine().split(**" "**);  
 **int** x=Integer.*parseInt*(str[0]);  
 **int** y=Integer.*parseInt*(str[1]);  
 **int** desx=Integer.*parseInt*(str[2]);  
 **int** desy=Integer.*parseInt*(str[3]);  
 *horse*(x,y,desx,desy,sizex,sizey);  
 sc.close();  
 }  
 **private static void** horse(**int** x,**int** y,**int** desx,**int** desy,**int** sizex,**int** sizey) {  
 **boolean**[][] visited=**new boolean**[1020][1020];  
 Queue<Node> queue=**new** ArrayDeque<>();  
 **int**[][] axes= {{1,2},{1,-2},{2,1},{2,-1},{-1,2},{-1,-2},{-2,1},{-2,-1}};  
 **for**(**int** i = 1;i<=8;i++){  
 **for**(**int** j = 1;j<=8;j++){  
 visited[i][j] = **false**;  
 }  
 }  
 Node node=**new** Node(x,y,0);  
 visited[x][y] = **true**;  
 queue.offer(node);  
 **while**(!queue.isEmpty()) {  
 Node front=queue.poll();  
 **int** a=front.**x**;  
 **int** b=front.**y**;  
 **int** step=front.**step**;  
 **if**(a==desx&&b==desy) {  
 System.***out***.println(step);  
 }**else** {  
 **for**(**int** i=0;i<axes.**length**;i++) {  
 **int** hx=a+axes[i][0];  
 **int** hy=b+axes[i][1];  
 **int** hstep=step+1;  
 Node ho=**new** Node(hx,hy,hstep);;  
 **if**(ho.**x**>=1&&ho.**x**<=sizex&&ho.**y**>=1&&ho.**y**<=sizey&&visited[ho.**x**][ho.**y**]==**false**) {  
 queue.offer(ho);  
 visited[ho.**x**][ho.**y**]=**true**;  
 }  
 }  
 }  
 }  
 }  
}  
**class** Node{  
 **public int x**;  
 **public int y**;  
 **public int step**;  
  
 **public** Node(**int** x,**int** y,**int** step){  
 **this**.**x** = x;  
 **this**.**y** = y;  
 **this**.**step**=step;  
 }  
}

**import** java.util.Scanner;  
  
**public class** Short  
{  
 **static int** *N*;*//城市个数* **static int** *M*;*//双向道路* **static int** *S*;*//起点* **static int** *D*;*//终点* **static int** *G*[][];*//邻接矩阵* **public static void** main(String[] args) {  
 *//* ***TODO Auto-generated method stub*** Scanner in = **new** Scanner(System.***in***);  
 System.***out***.println(**"请输入城市个数和双向道路条数"**);  
  
 *N* = in.nextInt();  
 **if** (*N*<0||*N*>100){  
 **return**;  
 }  
 *M* = in.nextInt();  
 **if** (*M*<1||*N*>(*N*\*(*N*-1))/2){  
 **return**;  
 }  
 *G* = **new int**[*N*][*N*];  
 **int** path[][] = **new int**[*N*][*N*];  
 **for**(**int** i = 0;i<*N*;i++)  
 **for**(**int** j = 0; j<*N*;j++) {  
 *G*[i][j] = Integer.***MAX\_VALUE***/100;  
  
 }  
  
 **for**(**int** i =0;i<*M*;i++) {  
 **int** x = in.nextInt();  
 **if**(x<1){  
 **return**;  
 }  
 **int** y = in.nextInt();  
 **if**(y>*N*){  
 **return**;  
 }  
 *G*[x][y] = in.nextInt();  
 **if**(*G*[x][y]>1000||*G*[x][y]<1){  
 **return** ;  
 }  
 *G*[y][x] = *G*[x][y];  
  
 }  
 System.***out***.println(**"请输入q次询问的次数："**);  
 **int** q=in.nextInt();  
 **for**(**int** l=0;l<q;l++){  
 System.***out***.println(**"请输入本次询问的起点和终点"**);  
 *S* = in.nextInt();  
 **if**(*S*<1){  
 **return**;  
 }  
 *D* = in.nextInt();  
 **if**(*D*>*N*){  
 **return**;  
 }  
  
 **for**(**int** k = 0;k<*N*;k++){  
 **for**(**int** i = 0; i<*N*;i++){  
 **for**(**int** j = 0;j<*N*;j++) {  
 **if**(i==j&&i==k&&j==k){  
 *G*[i][j]=0;}  
 **else if**(*G*[i][k]+*G*[k][j]<*G*[i][j]) {  
  
 *G*[i][j] = *G*[i][k]+*G*[k][j];  
  
 path[i][j] = k;  
 }  
  
 }}}  
 **if**(*G*[*S*][*D*]==Integer.***MAX\_VALUE***/100){  
 System.***out***.println(**"本次起点到终点的最短路径为："**+**"-1"**);  
 }**else** System.***out***.println(**"本次起点到终点的最短路径为："**+*G*[*S*][*D*]);  
  
 }  
  
  
 }  
  
}

**import** java.util.Scanner;  
**public class** EX4 {  
 **public void** lengthOfLIS(**int**[] nums){  
 **int** len = nums.**length**;  
 **if**(len < 2){  
 System.***out***.println(len-1);  
 }  
 **int**[] dp = **new int**[nums.**length**];  
 dp[0] = 1;  
 **int** maxLen = 0;  
 Scanner sc=**new** Scanner(System.***in***);  
 System.***out***.println(**"请输入样例总数："**);  
 **int** a=sc.nextInt();  
 **if**(a>200){  
 **return** ;  
 }  
  
 **for**(**int** l=0;l<a;l++){  
 System.***out***.println(**"请输入该组样例元素个数："**);  
 **int** b=sc.nextInt();  
 **if** (b>500){  
 **return**;  
 }  
   
*// for(int i = 0; i < a.length; i++){//��������  
// a[i] = sc.nextInt();  
// }  
//* System.***out***.println(**"请输入该组样例元素："**);  
 **for**(**int** i = 0; i <b; i ++){  
  
 nums[i] = sc.nextInt();  
 **int** maxVal = 0;  
 **for**(**int** j = 0; j < i; j ++){  
 **if**(nums[i] > nums[j]){  
 maxVal = Math.*max*(maxVal, dp[j]);  
 }  
 }  
 dp[i] = maxVal + 1;  
 maxLen = Math.*max*(maxLen, dp[i]);  
 }  
 System.***out***.println(**"修改最少元素个数为："**+(b-maxLen));  
   
  
 }}  
 **public static void** main(String[] args){  
 **int** nums[]=**new int**[1000];  
 EX4 ex=**new** EX4();  
   
 ex.lengthOfLIS(nums);  
   
 }  
}

#include <iostream>

#include <algorithm>

#include <stack>

#include<iostream>

using namespace std;

int knife; //切了多少刀

int stack\_number; //一共有几堆积木

int max\_H\_stack; //最大横堆有多少个积木

int arr[100]; //每堆积木多少个

int max = 0; // 最大值

int y;

int z;//发生替换的位置

int Max(int arr[])

{

int max=0;// 最大值

for (int i = 0; i < stack\_number;i++) {

if (arr[i] > max) { // 当前值大于最大值，赋值为最大值

max = arr[i];

y = i;

}

}

return max;

}

int look\_h\_max(int arr[])

{

max\_H\_stack = 0;

int temp=0;

for (int i = 0; i < stack\_number; i++) {

if (arr[i] >0) {

temp = temp + 1;

if (temp >= max\_H\_stack)

{

max\_H\_stack = temp;

z = i;

}

}

else {

temp = 0;

}

}

return max\_H\_stack;

}

int calculate\_knife(int arr[], int stack\_number)

{

bool flag=true;

int temp = 0;

while(temp!= stack\_number)

//for(int i=0;i<=10;i++)

{

if (Max(arr) > look\_h\_max(arr))

{

//cout << "纵切:" << Max(arr);

arr[y] = 0;

}

else

{

//cout << "横切:" << max\_H\_stack;

for (int i = z; i >=0; i--)

{

if (arr[i] > 0) {

arr[i] = arr[i] - 1;

}

else {

break;

}

}

}

knife = knife + 1;

temp = 0;

for (int i = 0; i < stack\_number; i++)

{

if (arr[i] <= 0) {

temp++;

}

}

}

return knife;

}

int main()

{

cin >> stack\_number;

for (int i = 0; i < stack\_number; i++) {

cin >> arr[i];

}

calculate\_knife(arr, stack\_number);

cout << knife;

return 0;

}