Given a string and the number of rows to arrange the string in zig-zag manner. Write a program to print the final string obtained after arranging characters in the string in zig-zag manner

For example:

If the given string is ABCDEFGH and the number of rows are 2 then the string obtained after zig-zag arrangement is

A C E G

B D F H

And the final resulting string is ACEGBDFH.

Code:

#include <stdio.h>

#include <string.h>

#include<stdlib.h>

int main() {

char str[50];

int key;

gets(str);

scanf("%d",&key);

int len = strlen(str);

if (key <= 1) {

printf("%s\n", str);

exit(0);

}

char result[len + 1];

int index = 0;

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

for (int j = i; j < len; j += 2 \* (key - 1)) {

result[index++] = str[j];

if (i != 0 && i != key - 1) {

int diag = j + 2 \* (key - i - 1);

if (diag < len) {

result[index++] = str[diag];

}

}

}

}

result[index] = '\0';

printf("%s\n", result);

return 0;

}

Write a program to insert an element into an array at any given position

#include <stdio.h>

void main()

{

int array[100];

int i, n, x, pos;

printf("Enter the number of elements in the array \n");

scanf("%d", &n);

printf("Enter the elements \n");

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

{

scanf("%d", &array[i]);

}

printf("Input array elements are: \n");

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

{

printf("%d ", array[i]);

}

printf("\nEnter the new element to be inserted: ");

scanf("%d", &x);

printf("Enter the position where element is to be inserted: ");

scanf("%d", &pos);

n=n+1;

for(i = n-1; i >= pos; i--)

array[i]=array[i-1];

array[pos-1]=x;

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

{

printf("%d ", array[i]);

}

}

Write a delete an element from an array from any given position

#include <stdio.h>

int main() {

int array[100], position, c, n;

printf("Enter number of elements in array\n");

scanf("%d", &n);

printf("Enter %d elements\n", n);

for (c = 0; c < n; c++)

scanf("%d", &array[c]);

printf("Enter the location where you wish to delete element\n");

scanf("%d", &position);

if (position >= n+1)

printf("Deletion not possible.\n");

else

{

for (c = position - 1; c < n - 1; c++)

array[c] = array[c+1];

printf("Resultant array:\n");

for (c = 0; c < n - 1; c++)

printf("%d\n", array[c]);

}

return 0;

}

Sort By Shift 1

One day, Sujith is interested in finding how to sort a sequence of integers a1, a2, ... an in non-decreasing order. Being a young kid, the only operation she can perform is a unit shift. That is, she can move the last element of the sequence to its beginning:

a1, a2, ..., an → an, a1, a2, ..., an - 1.

Help Sujith to calculate: What is the minimum number of operations that she needs to sort the sequence?

**Input and Output Format:**

The first line contains an integer, n (2 ≤ n ≤ 10^5).

The second line contains n integer numbers a1, a2, ..., an (1 ≤ ai ≤ 10^5).

If it's impossible to sort the sequence output -1. Otherwise output the minimum number of operations Sujith needs to sort it.

**Sample Input 1**

2

2 1

**Sample Output 1**

1

**Sample Input 2**

3

1 3 2

**Sample Output 2**

-1

**Sample Input 3**

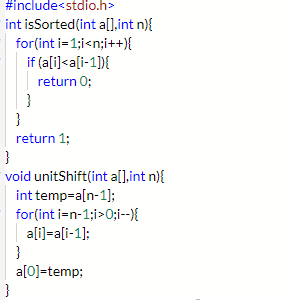
2

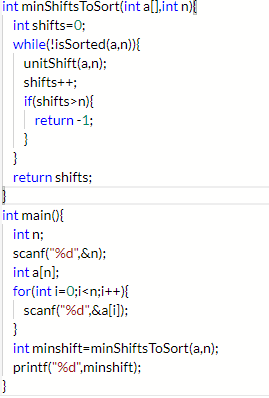
1 2

**Sample Output 3**

0

Code:





Antique Items

Neeraja is a great admirer of antique items and she has a huge collection of antique items at home. Once she wanted to expand her collection with exactly one antique item. Neeraja knows **n**sellers of antiques, the **i-th** of them auctioned **ki** items. Currently the auction price of the **j-th**object of the **i-th** seller is **sij**. Neeraja gets on well with each of the**n** sellers. She is perfectly sure that if she outbids the current price of one of the items in the auction (in other words, offers the seller the money that is strictly greater than the current price of the item at the auction), the seller of the object will immediately sign a contract with her.

Unfortunately, Neeraja has only **v** units of money. Help her to determine which of the n sellers she can make a deal with.

**Input and Output Format:**

The first line of the input contains two space-separated integers **n**, **v** (1 ≤ n ≤ 50;  v ≤ 1000000) — the number of sellers and the units of money Neeraja has.

Then **n** lines follow. The **i-th** line first contains integer **ki**(1 ≤ ki ≤ 50) the number of items of the **i-th** seller. Then go **ki** space-separated integers **si1, si2, ..., siki** ( sij≤ 1000000) — the current prices of the items of the **i-th** seller.

In the first line of the output, print integer **p** — the number of sellers with whom Neeraja can make a deal.

In the second line print **p** space-separated integers q1, q2, ..., qp (1 ≤ qi ≤ n) — the numbers of the sellers with whom Neeraja can make a deal. Print the numbers of the sellers in the increasing order.

**Sample Input 1**

3 50000

1 40000

2 20000 60000

3 10000 70000 190000

**Sample Output 1**

3

1 2 3

**Sample Input 2**

3 50000

1 50000

3 100000 120000 110000

3 120000 110000 120000

**Sample Output 2**

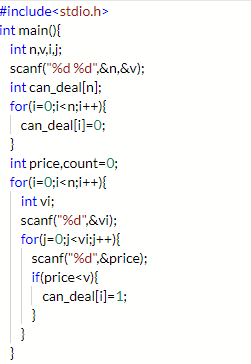
0

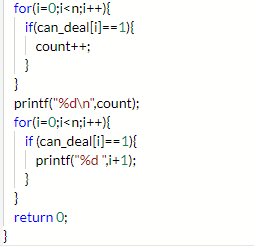
{**Hint:**

In the first sample Mahirl can bargain with each of the sellers. She can outbid the following items: a 40000 item from the first seller, a 20000 item from the second seller, and a 10000 item from the third seller.

In the second sample Mahirl cannot make a deal with any of the sellers, as the prices of all items in the auction are too big for her.}

Code:





Rita and Candies

There are n children in Rita's school. Rita is going to give some candies to them. Let's number all the children from 1 to n. The **i-th** child wants to get at least **ai** candies. Rita asks children to line up. Initially, the **i-th** child stands at the i-th place of the line. Then Rita starts distribution of the candies.

She follows the algorithm:

1. Give m candies to the first child in the line.
2. If this child still haven't got enough candies, then the child goes to the end of the line, else the child goes home.
3. Repeat the first two steps while the line is not empty.

Consider all the children in the order they go home. Rita wants to know, which child will be the last in this order?

**Input and Output Format**

The first line contains two integers n, m (1 ≤ n ≤ 100; 1 ≤ m ≤ 100). The second line contains n integers a1, a2, ..., an (1 ≤ ai ≤ 100).

Output a single integer, representing the number of the last child.

**Sample Input 1**

5 2

1 3 1 4 2

**Sample Output 1**

4

**Sample Input 2**

6 4

1 1 2 2 3 3

**Sample Output 2**

6

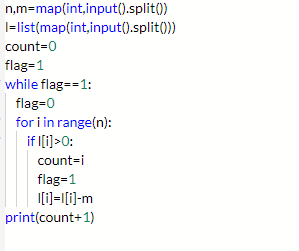
{**Hint :**

Let's consider the first sample.

First child 1 gets 2 candies and goes home. Then child 2 gets 2 candies and goes to the end of the line. Currently the line looks like [C3, C4, C5, C2] (indices of the children in order of the line). Then child 3 gets 2 candies and goes home, and then child 4 gets 2 candies and goes to the end of the line. Currently the line looks like [C5, C2, C4]. Then child 5 gets 2 candies and goes home. Then child 2 gets two candies and goes home, and finally child 4 gets 2 candies and goes home.

Child 4 is the last one who goes home.}

Code:



**Circular Array Rotation**

John Watson knows of an operation called a *right circular rotation* on an array of integers. One rotation operation moves the last array element to the first position and shifts all remaining elements right one. To test Sherlock's abilities, Watson provides Sherlock with an array of integers. Sherlock is to perform the rotation operation a number of times then determine the value of the element at a given position.

For each array, perform a number of right circular rotations and return the value of the element at a given index.

For example, array a = [3, 4, 5], number of rotations, k = 2 and indices to check, m = [1, 2].  
First we perform the two rotations:

[3, 4, 5] -> [5, 3, 4] -> [4, 5, 3]  
Now return the values from the zero-based indices 1 and 2 as indicated in the ***m,*** array.  
a[1] = 5

a[2] = 3

**Input Format**

The first line contains ***3*** space-separated integers, n, k, and q, the number of elements in the integer array, the rotation count and the number of queries.  
The second line contains ***n*** space-separated integers, where each integer ***i*** describes array element a[i] (where 0 <= i < n).  
Each of the ***q*** subsequent lines contains a single integer denoting ***m***, the index of the element to return from a.

**Constraints**

1 <= n <= 105

1 <= a[i] <= 105

1 <= k <= 105

1 <= q <= 500

0 <= m < n

**Output Format**

For each query, print the value of the element at index ***m*** of the rotated array on a new line.

**Sample Test Case 1**

**Input:**

3 2 3

1 2 3

0

1

2

**Output:**

2

3

1

**Explanation 0**

After the first rotation, the array becomes [3, 1, 2].  
After the second (and final) rotation, the array becomes [2, 3, 1].

Let's refer to the array's final state as array b = [2, 3, 1]. For each query, we just have to print the value of bm on a new line:

1. m = 0, b0 = 2.
2. m = 1, b0 = 3.
3. m = 2, b0 = 1.

**Sample Test Case 2**

**Input:**

4 3 3

1 2 3 4

2

1

0

**Output:**

4

3

2

**Hidden Test Case 1**

10 2 3

1 2 3 4 5 6 7 8 9 10

9

5

3

**Output:**

8

4

2

**Hidden Test Case 2**

2 6 1

1 2

0

**Output:**

1

**Hidden Test Case 3**

6 1 3

1 2 3 4 5 6

1

3

5

**Output:**

1

3

5

**Hidden Test Case 4**

10 2 5

0 1 10 100 1000 10000 2 4 6 8

2

4

6

8

9

**Output:**

0

10

1000

2

4

**Developed by:B. Vinay Kumar**

**Source: General**

**C Code:**

#include <stdio.h>

int main()

{

int n, k, q;

scanf("%d %d %d", &n, &k, &q);

int ar[n];

int qr[q];

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

scanf("%d", &ar[i]);

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

scanf("%d", &qr[i]);

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

printf("%d\n", ar[(n-(k%n)+qr[i])%n]);

return 0;

}

**Java Code:**

import java.util.\*;

class Main{

public static void main(String args[]){

Scanner sc=new Scanner(System.in);

int n = sc.nextInt();

int k = sc.nextInt();

int q = sc.nextInt();

int ar[] = new int[n];

int qr[] = new int[q];

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

ar[i] = sc.nextInt();

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

qr[i] = sc.nextInt();

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

System.out.println(ar[(n-(k%n)+qr[i])%n]);

return;

}

}

**Python Code:**

n, k, q = map(int,input().split())

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

qr = list()

for i in range(q):

qr.append(int(input()))

for i in range(q):

print(ar[(n-(k%n)+qr[i])%n])

**Centroid**

Write a program to find the centroid of an object in a 2-D grid.

A 2D grid is represented by values '0'(zeros) for background and by values '1'(ones) representing the object. Assume that the coordinates of the 1st point in the grid is (0,0). Assume that the 2D grid always consists of only one object surrounded by zeros as given below.  
  
The centroid is a float value pair (xc,yc) where xc is the average of all x coordinate values of all the coordinates belonging to the object and yc is the average of all y coordinate values of all the coordinates belonging to the object.

**Input and Output Format:**

Input consists of (m\*n) + 2 integers.

The 1st 2 integers are on 2 separate lines.

The 1st integer corresponds to m, the number of rows in the grid.

The 2nd integer corresponds to n, the number of columns in the grid.

The next 'n' integers correspond to the values in the 1st row.

The next 'n' integers correspond to the values in the 2nd row and so on.

Output consists of 2 float values that correspond to the x and y coordinate of the centroid. The 2 float values are separated by a space. The float values are printed correct to 2 decimal places.

**Sample Input:**

8

8

0 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 0  
0 0 1 1 1 1 0 0  
0 0 1 1 1 1 0 0  
0 0 0 1 1 0 0 0  
0 0 0 1 1 0 0 0  
0 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 0

**Sample Output:**

3.16,3.50

Code:

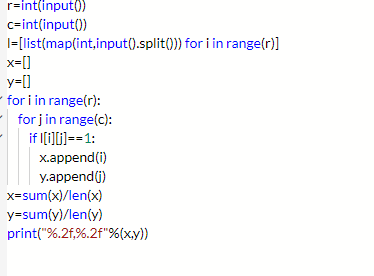


Image Enlarger

****  
**Problem Statement**

Given an image as a 2-D character array or a string [], scale it up by factor. That is, if the original image is M x N, the scaled image should be (M\*factor) x (N\*factor). Each character in the input should be represented by factor x factor of the same character in the output (see examples).  
  
**Constraints**  
- Range of M and N will be between 1 and 50 elements, inclusive.  
- Each character in image will have ASCII value between 32 and 126, inclusive.  
- Factor will be between 0 and 50, inclusive.  
- The output will contain at most 7500 characters.

**Input and Output Format:**

The first line of the input consists of an integer that corresponds to M.

The second line of the input consists of an integer that corresponds to N.

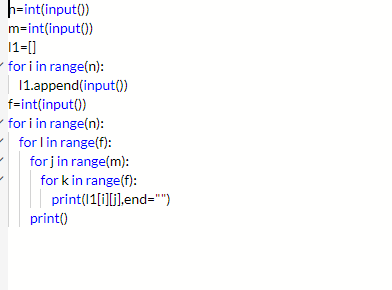
The next M lines of the input corresponds to the image.

The last line of the input consists of an integer that corresponds to the factor.

Output consists of the scaled up image.

Refer sample output for formatting specifications.  
  
**Sample Input:**  
3  
3  
abc  
def  
ghi  
3  
  
**Sample Output:**  
aaabbbccc  
aaabbbccc  
aaabbbccc  
dddeeefff  
dddeeefff  
dddeeefff  
ggghhhiii  
ggghhhiii  
ggghhhiii

Code:



Given a matrix print the elements diagonally downwards

#include <stdio.h>

int main()

{

int n;

scanf("%d", &n);

int arr[n][n];

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

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

scanf("%d",&arr[i][j]);

}

}

for (int k = 0; k < n; k++) {

int row = 0, col = k;

while (col >= 0) {

printf("%d ", arr[row][col]);

row++, col--;

}

}

for (int j = 1; j < n; j++) {

int col = n - 1, row = j;

while (row < n) {

printf("%d ",arr[row][col]);

row++, col--;

}

}

return 0;

}

Given a matrix print the elements diagonally upwards

#include <stdio.h>

int main()

{

int n;

scanf("%d", &n);

int arr[n][n];

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

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

scanf("%d",&arr[i][j]);

}

}

for (int k = 0; k < n; k++) {

int row = k, col = 0;

while (row >= 0) {

printf("%d ", arr[row][col]);

row--, col++;

}

}

for (int j = 1; j < n; j++) {

int col = j, row = n-1;

while (col < n) {

printf("%d ",arr[row][col]);

row--, col++;

}

}

return 0;

}

Write a program to print matrix elements in zigzag form

#include <stdio.h>

int main()

{

int m,n;

scanf("%d %d",&m, &n);

int arr[n][n];

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

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

scanf("%d",&arr[i][j]);

}

}

int evenRow = 0;

int oddRow = 1;

while (evenRow<m)

{

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

{

printf("%d ",arr[evenRow][i]);

}

evenRow = evenRow + 2;

if(oddRow < m)

{

for (int i=n-1; i>=0; i--)

{

printf("%d ",arr[oddRow][i]);

}

}

oddRow = oddRow + 2;

}

return 0;

}