# Exercises: Lists and Matrices

Problems for exercises and homework for the [“Programming Fundamentals” course @ SoftUni](https://softuni.bg/courses/programming-fundamentals).

You can check your solutions here: <https://judge.softuni.bg/Contests/208/Lists-and-Matrices-Exercises>

## Max Sequence of Equal Elements

Read a **list of integers** and find the **longest sequence of equal elements**. If several exist, print the **leftmost**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3 4 4 5 5 5 2 2 | 5 5 5 |
| 7 7 4 4 5 5 3 3 | 7 7 |
| 1 2 3 3 | 3 3 |

### Hints

* Scan positions **p** from left to right and keep the **start** and **length** of the current sequence of equal numbers ending at **p**

## Matrix of Palindromes

Write a program to generate the following **matrix of palindromes** of **3** letters with **r** rows and **c** columns.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 4 6 | aaa aba aca ada aea afa  bbb bcb bdb beb bfb bgb  ccc cdc cec cfc cgc chc  ddd ded dfd dgd dhd did |

## Diagonal Difference

Write a program that finds **absolute difference between the sums of square matrix diagonals**. On the **first line** you receive N – size of matrix. On the next **N lines,** you receive **values for every row** of the matrix **separated with space**.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 3  11 2 4  4 5 6  10 8 -12 | **15** | **Primary diagonal:** 11 + 5 + (-12) = 4  **Secondary diagonal:** -> 4 + 5 + 10 = 19  **Difference:** |4 – 19| = 15 |

## 2x2 Squares in Matrix

Find the count of 2x2 squares of equal chars in a matrix.

* The matrix size (**rows** and **columns**) is given at the first row
* Matrix characters come at the next **rows** lines (space separated)

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3 4  A B B D  E B B B  I J B B | 2 |
| 2 2  a b  c d | 0 |

## Max Platform

Write a program, which creates a rectangular array with size of n by m elements. The dimensions and the elements should be read from the console. Find a **platform with size of (3, 3)** **with a maximal sum**. On the **first line** of output **print the sum** and **then print the platform itself**. On First line of input you will receive number of rows and columns of the matrix separated with space.

### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 4 4  5 6 2 8  3 1 9 5  8 1 6 9  1 5 3 4 | **47**  **6 2 8**  **1 9 5**  **1 6 9** |  | 5 6  1 2 4 8 9 6  2 4 1 3 4 2  2 7 9 9 9 7  8 6 9 9 9 6  9 5 9 9 9 9 | **81**  **9 9 9**  **9 9 9**  **9 9 9** |

## Hourglass Sum

### You are given a 2D array with dimensions 6x6. An hourglass in an array is defined as a portion shaped like this:

|  |  |  |
| --- | --- | --- |
| **a** | **b** | **c** |
|  | **d** |  |
| **e** | **f** | **g** |

For example, if we create an hourglass using the number 1 within an array full of zeros, it may look like this:

1 1 1 0 0 0

0 1 0 0 0 0

1 1 1 0 0 0

0 0 0 0 0 0

0 0 0 0 0 0

0 0 0 0 0 0

Actually, there are many hourglasses in the array above. The three topmost hourglasses are the following:

1 1 1 1 1 0 1 0 0

1 0 0

1 1 1 1 1 0 1 0 0

The sum of an hourglass is the sum of all the numbers within it. The sum for the hourglasses above are 7, 4, and 2, respectively.

In this problem, you have to print the largest sum among all the hourglasses in the array.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| **1 1 1 0 0 0**  **0 1 0 0 0 0**  **1 1 1 0 0 0**  **0 0 2 4 4 0**  **0 0 0 2 0 0**  **0 0 1 2 4 0** | 19 |

## Matrices

Write a program, which creates **matrices** like those in the **figures below** and prints them formatted to the console. The size of the matrices will be read from the console. E.g. matrices with size of 4 x 4:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Type A:** | | | |  | **Type B:** | | | |  | **Type C:** | | | |  | **Type D\*:** | | | | |
| **1** | **5** | **9** | **13** |  | **1** | **8** | **9** | **16** |  | **7** | **11** | **14** | **16** |  | **1** | **12** | **11** | **10** |
| **2** | **6** | **10** | **14** |  | **2** | **7** | **10** | **15** |  | **4** | **8** | **12** | **15** |  | **2** | **13** | **16** | **9** |
| **3** | **7** | **11** | **15** |  | **3** | **6** | **11** | **14** |  | **2** | **5** | **9** | **13** |  | **3** | **14** | **15** | **8** |
| **4** | **8** | **12** | **16** |  | **4** | **5** | **12** | **13** |  | **1** | **3** | **6** | **10** |  | **4** | **5** | **6** | **7** |

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| A 4 4 | 1 5 9 13  2 6 10 14  3 7 11 15  4 8 12 16 |
| B 3 2 | 1 6  2 5  3 4 |
| C 4 5 | 7 11 15 18 20  4 8 12 16 19  2 5 9 13 17  1 3 6 10 14 |
| D 3 4 | 1 10 9 8  2 11 12 7  3 4 5 6 |

## \* Largest Frame in Matrix

Find the **largest rectangular frame of same number** in **matrix of integers**. If several equal sized largest frames exist, **print all of them**, **sorted alphabetically**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3 7  3 5 5 5 3 3 3  3 5 0 5 3 0 3  3 5 5 5 3 3 3 | 3x3, 3x3 |
| 3 6  7 7 0 8 8 8  7 7 0 8 8 8  7 7 7 7 7 7 | 1x6, 2x3, 3x2 |

## \*\* Largest Increasing Subsequence

Read a **list of integers** and find the **longest increasing subsequence**. If several such exist, print the **leftmost**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| **1** | 1 |
| 7 **3 5** 8 -1 0 **6 7** | 3 5 6 7 |
| **1 2** 5 **3 5** 2 4 1 | 1 2 3 5 |
| **0** 10 20 30 30 40 **1** 50 **2 3 4 5 6** | 0 1 2 3 4 5 6 |
| 11 12 13 **3** 14 **4** 15 **5 6 7 8** 7 **16** 9 8 | 3 4 5 6 7 816 |

### Hints

* This problem can be solved using dynamic programming

## Array Manipulator

Write a program that **reads an array of integers** from the console and **set of commands** and **executes them on the array**. The commands are as follows:

* **add <index> <element>** - adds element at the given index
* **addMany <index> <element 1> <element 2> … <element n>** - adds set of elements at the given index
* **contains <element>** - prints index of the element if exists in the array or -1 if the element is not found
* **remove <index>** - removes element at given index
* **shift <positions>** - **shifts every element** of the array the number of positions **to the** **left**.

For example, [1, 2, 3, 4, 5] -> shift 2 -> [3, 4, 5, 1, 2]

* **sumPairs** -> sums elements in the array of pairs. For example, [1, 2, 4, 5, 6, 7] -> [3, 9, 13]
* **print** – stop receiving more commands and print the last state of the array.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1 2 4 5 6 7  add 1 8  contains 1  contains -3  print | 0  -1  [1, 8, 2, 4, 5, 6, 7] |
| 1 2 3 4 5  addMany 5 9 8 7 6 5  contains 15  remove 3  shift 1  print | -1  [2, 3, 5, 9, 8, 7, 6, 5, 1] |
| 2 2 4 2 4  add 1 4  sumPairs  print | [6, 6, 6] |
| 1 2 1 2 1 2 1 2 1 2 1 2  sumPairs  sumPairs  addMany 0 -1 -2 -3  print | [-1, -2, -3, 6, 6, 6] |

## Sum Reversed Numbers

Write a program that reads sequence of numbers reverses them and print their sum.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| **123 234 12** | 774 | 321 + 432 + 21 = 774 |
| 12 12 34 84 66 12 | 220 | 21 + 21 + 43+ 48 + 66 + 21 = 220 |
| 120 1200 12000 | 63 | 21 + 21 + 21 = 63 |

## Bomb Numbers

Write a program that **reads sequence of numbers** and **special bomb number** with a certain **power**. Your task is to **detonate every occurrence of the special bomb number** and according to its power **his neighbors from left and right**. Finally print the **sum of the remaining elements** of the sequence.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 1 2 2 4 2 2 2 9  4 2 | 12 | Special number is 4 with power 2. After detontaion we left with the sequence [1, 2, 9] with sum 12. |
| 1 4 4 2 8 9 1  9 3 | 5 | Special number is 9 with power 3. After detontaion we left with the sequence [1, 4] with sum 5. Since the 9 has only 1 neighbour from the right we remove just it. |
| 1 7 7 1 2 3  7 1 | 6 | Here we could not detonate the second occurance of 7 because its already destroyed from the first occurance |
| 1 1 2 1 1 1 2 1 1 1  2 1 | 4 |  |

## Tour

You are given a square matrix representing **distances between cities**. City names are numbers from 0 to N. Also you are a given a sequence of cities that needs to be visited in that order. You start from city 0. Your task is to write a program that reads the distances matrix, the sequence of numbers representing number of cities that will be visited starting from city 0 and finally print the total distance of the route. For example:

|  |  |  |  |
| --- | --- | --- | --- |
| **City** | **0** | **1** | **2** |
| **0** | 0 | 10 | 20 |
| **1** | 10 | 0 | 55 |
| **2** | 20 | 55 | 0 |

Distances between city 0 and city 1 is 10, between city 0 and city 2 is 20 and between 1 and 2 is 55.

### Examples

|  |  |  |
| --- | --- | --- |
| Input | Output | Comments |
| 3  0 10 20  10 0 55  20 55 0  1 0 2 1 | 95 | 3 is the size of the matrix (3 rows and 3 cols)  Starting from city 0. The route is as follows:  0 -> 1 -> 0 -> 2 -> 1  0 -> 1 (distance 10)  1 -> 0 (distance 10)  0 -> 2 (distance 20)  2 -> 1 (distance 55)  Total distance: 95 |