Homework: C Arrays

This document defines the homework assignments from [the "C Programming" Course @ Software University]. Please submit as homework a single **`zip`** / **`rar`** / **`7z`** archive holding the solutions (source code) of all below described problems.

Problem 1. Save and Print Numbers in Range

Write a program that reads **n** numbers from the console and saves them in an array. The program should then print the elements of the array on the console.

Input	Output
5 3 2 4 5 8	3 2 4 5 8

Input	Output
3 2 1 -3	2 1 -3

Problem 2. Linear Search

Given an array of **n** integers, write a **linear search function** that determines whether a given element exists in the array. On the first line you will receive the number **n**. On the second line, there will be **n** numbers, space-separated. On the third line, the search number will be given.

Input	Output
5 3 9 8 7 1 3	yes

Input	Output
9 1 -9 10 3 2 -10 13 4 9 -3	no

Problem 3. Sort Array of Numbers

Write a program to read an array of numbers from the console, **sort them** and print them back on the console. Use a **sorting algorithm of your choosing**. The numbers should be entered one at a line. On the first input line you will be given the count of the numbers. Examples:

Input	Output
7 6 5 4 10 -3 120 4	-3 4 4 5 6 10 120

Problem 4. Categorize Numbers and Find Min / Max / Average

Write a program that reads N **floating-point numbers** from the console. Your task is to separate them in two sets, one containing only the **round numbers** (e.g. 1, 1.00, etc.) and the other containing the **floating-point numbers with non-zero fraction**. Print both arrays along with their minimum, maximum, sum and average (rounded to two decimal places). The numbers should be entered one at a line. On the first input line you will be given the count of the numbers. Examples:

Input	Output

7 1.2 -4 5.00 12211 93.003 4 2.2	[1.2, 93.003, 2.2] -> min: 1.2, max: 93.003, sum: 96.403, avg: 32.13 [-4, 5, 12211, 4] - > min: -4, max: 12211, sum: 12216, avg: 3054.00

Problem 5. Longest Increasing Sequence

Write a program to find all **increasing** sequences inside an array of integers. The numbers should be entered one at a line. On the first input line you will be given the count of the numbers. Print the sequences in the order of their appearance in the input array, each at a single line. Separate the sequence elements by a space. Find also the longest increasing sequence and print it at the last line. If several sequences have the same longest length, print the **left-most** of them. Examples:

Input	Output
9 2 3 4 1 50 **2** **3** **4** **5**	2 3 4 1 50 2 3 4 5 Longest: 2 3 4 5
8 **8** **9** 9 9 -1 5 2 3	8 9 9 9 -1 5 2 3 Longest: 8 9
9 **1** **2** **3** **4** **5** **6** **7** **8** **9**	1 2 3 4 5 6 7 8 9 Longest: 1 2 3 4 5 6 7 8 9
6 5 **-1** **10** **20** 3	5 -1 10 20 3 4 Longest: -1 10 20

4	
10 **10** 9 8 7 6 5 4 3 2 1	10 9 8 7 6 5 4 3 2 1 Longest: 10

Problem 6. Join Lists

Write a program that takes as input two arrays of integers and **joins them**. The result should hold all numbers from the first array, and all numbers from the second array, **without repeating numbers**, and arranged in **increasing order**. On the first input line, you are given the count of the elements of the first array. On the next line you are given the length of the second array. Examples:

Input	Output
6 **5** 20 40 10 10 30 80 **25** **20** **40** **30** **10**	10 20 25 30 40 80
5 **3** 5 4 3 2 1 **6** **3** **2**	1 2 3 4 5 6
1 1 1 1	1

Problem 7. Reverse Array

Write a program that reverses an array of numbers. The numbers should be entered one at a line. On the first input line you will be given the count of the numbers.

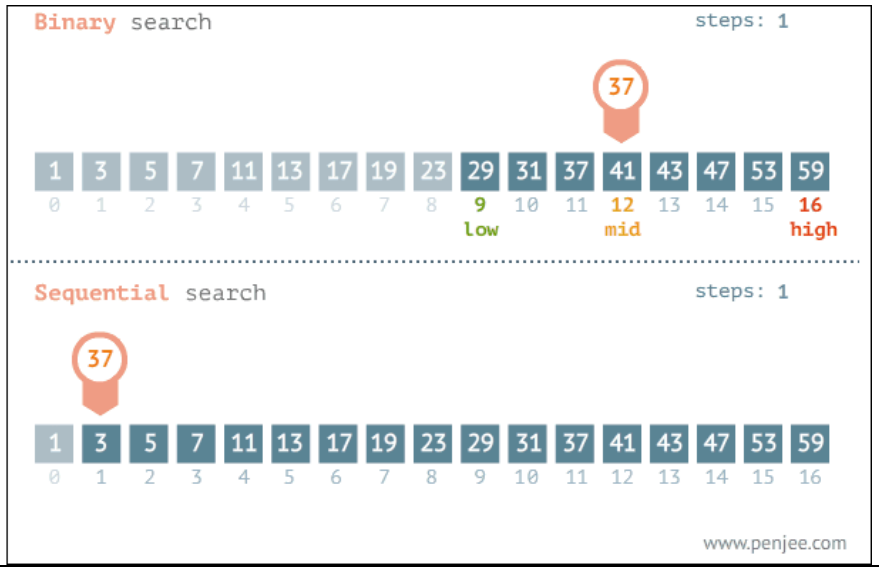
5 6 2 4 8 9	9 8 4 2 6
4 1 2 3 4	4 3 2 1

Problem 8. Iterative Binary Search

Binary search works only on **sorted collections**. It picks the **mid element** of the collection and checks if it's equal to the searched element.

- If it's **equal**, returns the mid index.
- If it's **smaller**, cuts the right half of the collection and repeats the same step.
- If it's **larger**, cuts the left half of the collection and repeats the same step.

Linear Search vs Binary Search Comparison



Animated gif:
https://blog.penjee.com/wp-content/uploads/2015/04/binary-and-linear-search-animations.gif

The first line holds the number of elements. The second line contains all elements, space-separated. The third line holds the number we search for.

Input	Output
8 -2 0 3 5 213 8582 239191 985128 239191	6
10 0 1 2 3 4 5 6 6 7 8 -2	-1
8 3 9 10 12 13 13 13 13 13	4 (or 5)

Problem 9. * Recursive Binary Search

Implement binary search using **recursion**. You are not allowed to use loops.

Problem 10. Numbers Beneath Main Diagonal

You are given a matrix of numbers. Your task is to print out the numbers, one group at a line, which are stationed beneath the matrix's main diagonal along with the diagonal itself. On the first input line, you are given the matrix's rows and cols count.

Input	Output
4 **1** 2 3 4 5 **6** 7 8 9 10 **11** 12 13 14 15 **16**	**1** 5 **6** 9 10 **11** 13 14 15 **16**
6 **7** 8 1 13 20 11 6 **12** 15 67 34 56 2 1 **23** 24 35 19 3 4 15 **16** 45 78 4 1 55 88 **12** 99 8 14 66 67 90 **91**	**7** 6 **12** 2 1 **23** 3 4 15 **16** 4 1 55 88 **12** 8 14 66 67 90 **91**

Problem 11. Scalar Multiplication of a Vector

A Scalar Multiplication of vectors is when you have a given vector (ex. $[a_1, a_2 \ldots a_n]$) and you want to multiply it by a scalar (a simple number). The multiplication is done by multiplying the scalar with each of the vector members. Write a program to perform a scalar multiplication of a vector. You are given an input number representing the dimension of a vector on the first input line. On the second input line, you are given the scalar to multiply the vector and on the next n lines, you are given the members of the vector.

Input	Output

3 **2** 5 6 7	10 12 14
2 **4** 12 14	48 56

Problem 12. Dot Product of vectors

A Dot product of two vectors is a scalar which is a result of the sum of the product of each of the two vectors' members. For example, if we have the vector $[a_1, a_2 \ldots a_n]$ and the vector $[b_1, b_2 \ldots b_n]$, their dot product is: $(a_1b_1 + a_2b_2 \ldots + a_nb_n)$. You are given the length **n** of each of the vectors on the first input line (In order to perform a dot product of two vectors, they have to be with equal dimensions). On the next n lines, you are given the members of the first vector, and on the next n ones, the members of the second one.

Input	Output
3 1 2 3 **4** **5** **6**	32 Explanation: 1 * 4 + 2 * 5 + 3 * 6 = 4 + 10 + 18 = 32
2 5 6 **9** **14**	129

Problem 13. *Cross Product of Vectors

The cross product of vectors is defined only in a dimension where n = 3. It takes as input two vectors and produces a new one in the same dimension. Each member of the resulting vector is a result of the determinant of the members from different rows in the first vector. Example:

$[a_1, a_2, a_3] \times [b_1, b_2, b_3] = [\ (a_2*b_3 - a_3*b_2),\quad (a_3*b_1 - a_1*b_3),\quad (a_1*b_2 - a_2*b_1)\ ]$

Note how the middle row is calculated with the terms backwards or you could just take the negative of the regular calculation. On the first 3 input lines, you are given the members of the first vectors, and on the next ones, you are given the members of the second one. Examples:

Input	Output
1 2 3 **4** **5** **6**	[-3, 6, -3] Explanation: [(2*6 - 5*3), (4*3 - 1*6), (1*5 - 2*4)]

5 6 8 **9** **14** **5**	[-82, 47, 16]

Problem 14. Sum of Matrices

You are given two matrices and you have to output a new matrix which is their sum. The sum of matrices is calculated by adding each of the members from the first matrix with each of the members of the second one and producing a number which is a member of a new matrix and is stationed in the same row and col. Write a program to calculate the sum of two matrices. On the first 2 lines, you are given the dimensions of the two matrices (In order for them to be added together, they must be of the same dimensions). The first line represents the rows of the matrices and the second one – the columns. On the next input lines, you are given the members of the matrices. Examples:

Input	Output
2 3 5 6 7 1 2 3 **2 8 4** **3 5 5**	7 14 11 4 7 8
3 3 6 4 2 9 5 8 7 3 1 **10 4 13** **21 -3 -7** **4 8 6**	16 8 15 30 2 1 11 11 7

Problem 15. Multiplication of Matrices

A multiplication of two matrices is performed by taking each of the rows of the first matrix and taking the dot product of it and each of the columns in the second matrix. The newly created matrix has dimensions **RxC**, where **R** is the number of rows of the first matrix and **C** is the number of columns of the second matrix.

Note that a matrix multiplication is valid only if the number of rows of the first matrix is equal to the number of columns of the second matrix and the number of columns of the first one are equal to the number of rows of the second one. Example:

$$\begin{vmatrix} a_1, a_2, a_3 \\ a_4, a_5, a_6 \end{vmatrix} \times \begin{vmatrix} b_1, b_2 \\ b_4, b_5 \\ b_7, b_8 \end{vmatrix} = \begin{vmatrix} (a_1*b_1 + a_2*b_4 + a_3*b_7), & (a_1*b_2 + a_2*b_5 + a_3*b_8) \\ (a_4*b_1 + a_5*b_4 + a_6*b_7), & (a_4*b_2 + a_5*b_5 + a_6*b_8) \end{vmatrix}$$

On the first two input lines you are given the dimensions of the first matrix. The second matrix has the same dimensions in inversed order. Examples:

Input	Output
2	70 126

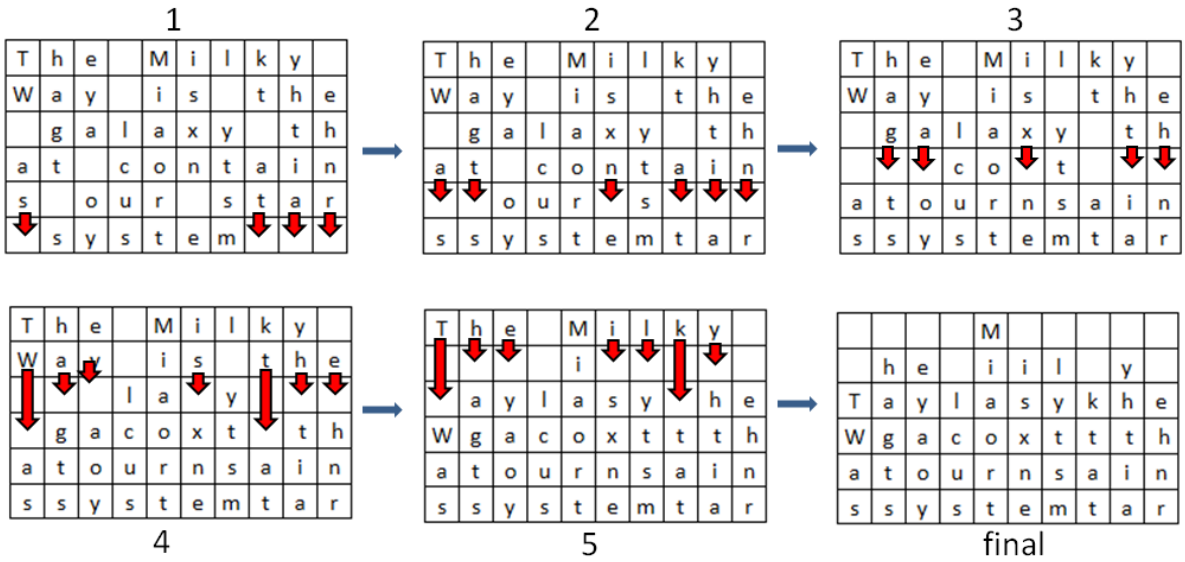
3 5 6 7 1 2 3 **2 8** **3 5** **6 8**	26 42 Explanation: | (5*2 + 6*3 + 7*6), (5*8 + 6*5 + 7*8) | | (1*2 + 2*3 + 3*6), (1*8 + 2*5 + 3*8) |
3 4 4 8 13 2 7 3 4 14 4 -5 -7 2 **-6 -8 2** **16 15 1** **-7 -2 4** **5 9 17**	23 80 102 48 107 271 -45 -75 9

Problems for Champions

Problem 16. Text Gravity

Write a program that takes as input a **line length** and **text** and formats the text so that it fits inside several rows, each with length equal to the given line length. Once the text is fitted, each character starts dropping as long as there is an empty space below it.

For example, we are given the text "*The Milky Way is the galaxy that contains our star system*" and **line length** of **10**. If we distribute the text characters such that the **text fits in lines with length 10**, the result is:



Text characters start 'falling' **until no whitespace remain under any character**. The resulting text should be printed in a table manner as shown below.

Input

The input will come from the console. It will consist of two lines.

- The first line will hold the **line length**.
- The second input line will hold a **string**.

Output

The output consists of lines, enclosed in pipe characters ('**|**'). Print **space** " " in all empty cells. See the example below.

Constraints

- The **line length** will be an integer in the range [1 ... 30].
- The **text** will consist of [1 … 1000] ASCII characters.

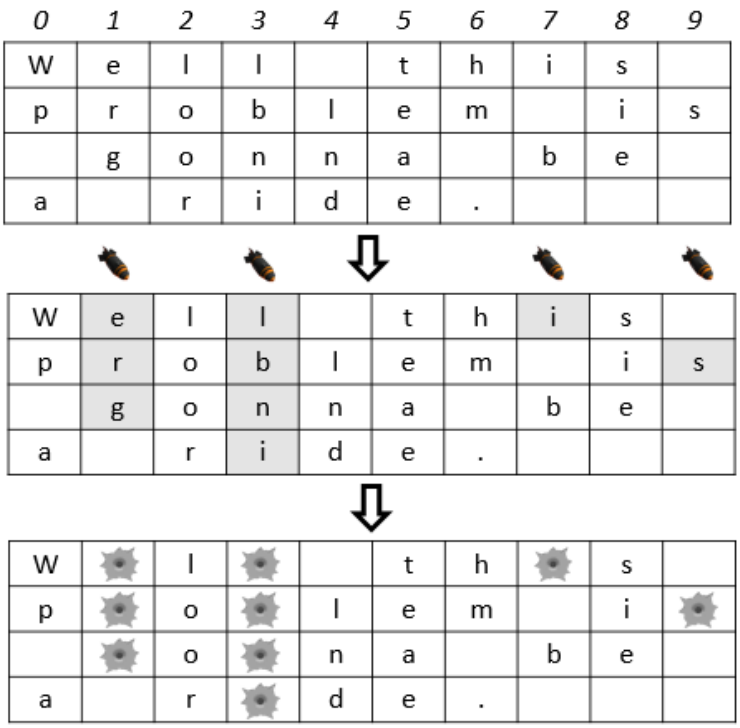
Example

Input
10 The Milky Way is the galaxy that contains our star system
Output

`|    M     |`
`| he iil y |`
`|Taylasykhe|`
`|Wgacoxttth|`
`|atournsain|`
`|ssystemtar|`

Problem 17. Text Bombardment

Write a program that reads a **text** and **line width** from the console. The program should distribute the text so that it fits in a table with a specific line width. Each cell should contain only **1 character**. It should then read a **line with numbers**, holding the **columns that should be bombarded**.



For example, we read the text "**Well this problem is gonna be a ride.**" and line width **10**. We distribute the text among 4 rows with 10 columns. We read the numbers "**1 3 7 9**" and drop bombs on those columns in the table.

The bombs **destroy** the character they fall on + all the neighboring characters **below** it. **Note**: Empty spaces below destroyed characters stop the bombs (see column 7).

Finally, we print the bombarded text on the console: "**W l th s p o lem i o na be a r de.**"

Note: The empty cells in the table after the text should NOT be printed.

Input

The input data is read from the console.

- On the first line you will be given the **text**
- On the next lines you will be given the **line width**
- On the third line you will receive the **columns** that should be bombed (space-separated)

The input data will always be valid and in the format described. There is no need to check it explicitly.

Output

The output data must be printed on the console and should contain only 1 line: the **bombarded text** as a single string.

Constraints

- The text will contain only ASCII characters and will be no longer than 1000 symbols.
- The line width will be in the range [1…100].
- The columns will be valid integers in the range [1…<line width> - 1].
- A column will not be bombed more than once.
- Time limit: 0.25 seconds. Allowed memory: 16 MB.

Example

Input	Output
Well this problem is gonna be a ride. 10 1 3 7 9	W l th s p o lem i o na be a r de.