Problems

# 1.

From SI read characters until you read the char !. In these characters are hidden integers (smaller than 100). Write a program that will read all characters and print the sum of all integers hidden in the text.

**For example:**

| **Input** | **Result** |
| --- | --- |
| ako34D neka12em bashka41mewr! | 87 |

# 2.

From SI read N strings not longer than 80 chars. At the beginning of the program read two integers:

* N - number of strings you are going to read
* X - shifting size.

Each of the read strings should be transformed in such a way that lowercase and uppercase letters (a-z, A-Z) are replaced with the same letter shifted X places afterwards in the alphabet (a-z). If you overflow the range of alphabet, continue cyclic from the beginning of the alphabet. Transformed string should be printed on SO.

**Transformation of a string should be implemented in a separate recursive function.**

Example:

Welcome -> transformed with shift 5 -> Bjqhtrj

# 3.

Write a program that will print the contents of the text file sifra.txt coded. The coding is done letter by letter, so that each letter (only letters, not digits, dots, ...) is replaced with the letter 3 places after in the alphabet. In case of the letters at the end of the alphabet, the counting continues cyclic from the beginning of the alphabet. In the final coded text all letters are lowercase. Example letter a is coded to d, H is coded to k, p is coded to s, x is coded to a, etc.

**For example:**

| **Input** | **Result** |
| --- | --- |
| Momentalno go testiram resenieto na mojata treta zadaca! | prphqwdoqr jr whvwludp uhvhqlhwr qd prmdwd wuhwd cdgdfd! |

# 4.

Let A = {a*1, a*2, .. a\_n} is an array of integers. The size of the array *n* and the elements of the array are read from SI. Write a program that will transform the array so each element of the original array is replaced with the number of appearance of the least significant digit (the right most) in the number itself. Print the result array on the standard output.

Compute the count of given digit in a number with separate **recursive** function.

Example:

Input:

5

1 11 1121 111222112 22222

Output:

1 2 3 4 5

**For example:**

| **Input** | **Result** |
| --- | --- |
| 3  18181818 900003 505 | 4 1 2 |

# 5.

Write a program that will read unknown number of three integers, and for each triple (a, b, c) will print the number (a or b) that contains the digit c more times on even positions (the positions are counted from right to left, and the first position is 1).

**For example:**

| **Input** | **Result** |
| --- | --- |
| 5436666 542456 6  1232432 3435 2  121211 22222 1  3333 4444 3  555514 1212 1  889999 99 9  50 20 2  81 21 8 | 5436666  1232432  121211  3333  1212  889999  20  81 |

# 6.

In a given file **"numbers.txt"** are written more rows with integers, and each row starts with one integer (*N >= 1*) that represents how many integers are following afterwards in the same row. Write a program that on SO will print print the number with largest most significant digit. Reading of integers ends when 0 is read.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 14 133944 278216 387399 28307 253543 419078 35197 211333 324609 260197 330161 395298 28991 59686  7 337422 459187 348310 78139 399034 456374 309499  15 471882 16606 52710 20269 202735 32904 132885 145244 98324 224723 311241 486996 2199 394455 171800  6 105192 290533 184513 488146 171447 479947  14 291296 136991 424759 479132 477586 61427 490850 356371 305920 159396 82301 467786 343436 201413  13 378398 301708 146052 422210 143716 400158 146357 41139 243123 471720 156271 452286 195993  3 333401 371831 377200  0 | 59686  78139  98324  488146  82301  422210  333401 |

# 7.

Write a program that reads matrix with dimensions MxN (max. 100x100). At the beginning the dimensions are read, and then the elements of the matrix with values 0 or 1. Your program should count and print on SO the number of rows and columns with at least 3 consecutive appearance of elements with value 1.

Example:

111100110011111010111001

1 row + 1 column = 2

**For example:**

| **Input** | **Result** |
| --- | --- |
| 10 4  0 0 0 1  0 1 0 0  0 0 0 1  1 0 0 0  0 0 0 1  1 0 0 1  0 1 0 0  1 0 1 1  0 1 1 0  1 0 1 1 | 1 |

# 8.

Read from standard input the dimensions of a matrix (m < 100 and n < 100) which contains only zeros and ones. Then read the elements of the matrix.

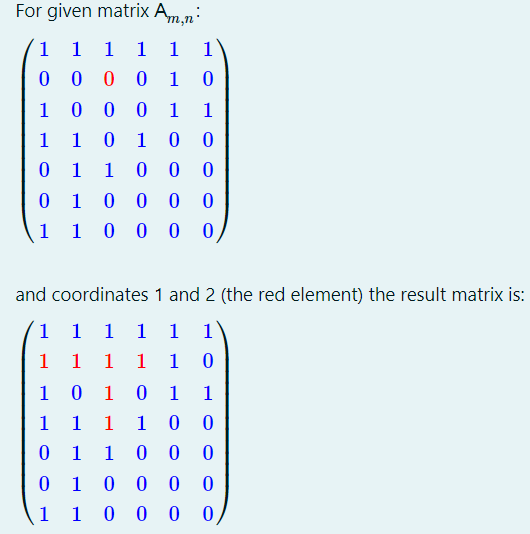
Read the coordinates of one element of the matrix. If the element of the read position is 0, then it should become 1 and:

* all zeros above that element to the first element with value 1 or to the start of the column,
* all zeros bellow that element to the first element with value 1 or to the end of the column,
* all zeros left from that element to the first element with value 1 or to the begin of the row and
* all zeros right from that element to the first element with value 1 or to the end of the row.

If the element of the given position is 1 than the matrix stays unchanged.

Print the changed matrix.

**Example:**



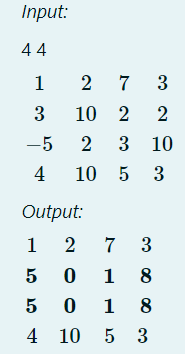
**For example:**

| **Input** | **Result** |
| --- | --- |
| 7 6  1 1 1 1 1 1  0 0 0 0 1 0  1 0 0 0 1 1  1 1 0 1 0 0  0 1 1 0 0 0  0 1 0 0 0 0  1 1 0 0 0 0  5 5 | 1 1 1 1 1 1  0 0 0 0 1 0  1 0 0 0 1 1  1 1 0 1 0 1  0 1 1 0 0 1  0 1 1 1 1 1  1 1 0 0 0 1 |

# 9.

The dimensions of one matrix (m,n <= 100), and then the elements of the matrix are entered from the keyboard. Make a transformation of the matrix so that the middle element in each column is replaced by the difference (by absolute value) of the sum of the elements in the first half of the column and the sum of the elements in the second half of the column. If the matrix has even number of rows, the value of the middle **two** elements should be changed. The middle element(s) are included when calculating the sum (in the case of an odd number of rows, the middle element is included when calculating both of the sums (upper and lower sum)!). Print the transformed matrix on the standard output.

**Example**

**

**For example:**

| **Input** | **Result** |
| --- | --- |
| 6 5  0 -5 0 -100 -10  0 0 0 -30 0  0 0 0 -20 22  0 0 45 0 0  0 4 -23 0 14  0 -2 11 0 0 | 0 -5 0 -100 -10  0 0 0 -30 0  0 7 33 150 2  0 7 33 150 2  0 4 -23 0 14  0 -2 11 0 0 |

# 10.

Write a **recursive function** that will find the number of odd numbers of an array of integers. The function accepts as an arguments the array for which the number of odd numbers is required and the total number of elements in the array. The function is given with the following prototype:

int Odd (int a [], int n);

Also write the main () function to test the Odd function.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 25  -1  -2  -5  5  7  -7  5  100  -6  -2  6  -6  -2  -9  -110  -24  -55  2  4  6  8  10  17  24  -25 | 10 |

# 11.

For one integer b we say that is a **parity complement** of other integer a **if and only if** in the number b all the even digits of the number a are replaced with the corresponding (odd) digit increased by 1, and all odd digits of the number a are replaced with the corresponding (odd) digit decreased by 1 .

**Example**. Number b = 4013698 is a parity complement of a = 5102789.

Read from SI unknown number of integers (not more than 100), until something that is not a number is entered.

Your task is to print the smallest 5 of the parity complements of all numbers, from the smallest to the largest.

**Notice**: If less then 5 numbers are entered, then print all the entered numbers.

Finding the parity complement of a number should be implemented in a separate **recursive function** parcom(int a).

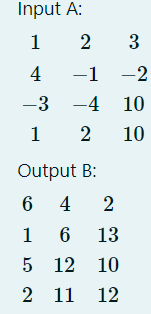
**Example**.

For numbers: 9641, 69403, 103007, 6, 11, 181, 20486 you should find their parity complements (those are: 8750, 78512, 12116, 7, 0, 90 and 31597, respectively), and print the 5 smallest of them in this order: 0 7 90 8750 12116.

Using global variables is **FORBIDDEN**.

# 12.

Write a program where a matrix *A* read from SI (maximum value of dimensions of the matrix is 100) is transformed to a new matrix *B*. Each element of the new matrix *B* is sum of positive neighbors of the corresponding element of the matrix *A*. Print the new matrix *B* on the standard output.



**For example:**

| **Input** | **Result** |
| --- | --- |
| 2 3  1 -5 -9  -2 3 8 | 0 4 8  4 8 3 |

# 13.

Write a program that will read a string (not longer than 100 characters) containing **letters and digits**. The program should print on the standard output the letters that are between the **first and the second digit**, left-to-right, including the bordering digits themselves. If there is only one digit in the string, the program should print all the letters till the end of the string. If there is no digits in the string the program should print out the message "No digits".

Example:

Axvdf7dbb3bdbd88B -> 7dbb3

Isdv88mvndkv9d -> 88

vnjis9kjsnvnv -> 9kjsnvnv

dsfsrtt -> No digits

**For example:**

| **Input** | **Result** |
| --- | --- |
| Isdv88mvndkv9d | 88 |

# 14.

An array of integers is to be read from SI. First the number N (N<=100) is entered followed by N integers. Write a program that will transform the array in the following way: starting at the beginning of the array, for each element that is equal to the next element in the array, double the value of the first one and overwrite the second one with 0. After that, rearrange the array so that all the zeroes will be moved at the end of the array, keeping the order of the other elements. Print all the elements of the array on the SO in a single line, separated by space.

Write **separate functions** for transforming the array and for reordering the elements.

Example:

6

2 2 0 4 8 8

(*Intermediate:* 4, 0, 0, 4, 16, 0)

Output:

4 4 16 0 0 0

**For example:**

| **Input** | **Result** |
| --- | --- |
| 7  0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 |

# 15.

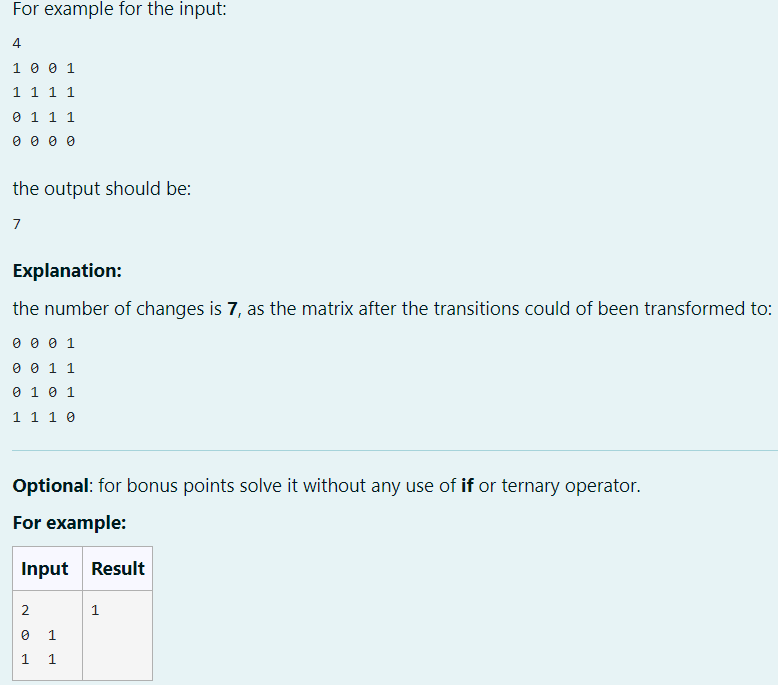
Read a square matrix of integers from standard input. First the dimension N is entered (2 < N <= 100) and then the elements of the matrix by rows.

Transform the matrix in a way that all the elements ABOVE the main diagonal will be set to the value of the maximal element found above the main diagonal, while all the elements BELOW the main diagonal will be set to the value of the minimal element found below the main diagonal. Print out the transformed matrix.

# 16.

Read a square binary matrix from standard input. First the dimension N is entered (1 < N <= 100) and then the elements of the matrix (either 0 or 1) by rows.

Write a program that will calculate the smallest number of necessary changes of the elements in order to transform the matrix to zero diagonal symmetrical matrix. Any transition of an element form 0 to 1 or form 1 to 0 is counts as a single change. After the conversion all the elements of the main diagonal should be 0, and the matrix should be symmetrical with respect to the main diagonal. Print the minimal number of transitions required.



# 17.

You are given a text file text.txt containing multiple lines of text, none of which is longer than 80 characters. Write e program that will transform the input file and print it out on the standard output. The output should contain only letters (all other characters are discarded), and each word should be printed in a separate line.

Example:

Input file:

Voi5 ovaa 6d6ato$$tek!a ke

ostanat09856 4453 sa5m2o

bu!!4kv(((((i

Output:

Vo

ovaa

datoteka

ke

ostanat

samo

bukvi

**For example:**

| **Input** | **Result** |
| --- | --- |
| Test two words | Test  two  words |

# 18.

You are given a text file input.txt containing a single word in each line of the file. No words are longer than 20 characters. Write a program that on the SO will print all the words from the file that are unsigned hexadecimal numbers (one word per line in uppercase digits). At the end also print the total number of such numbers found in the file.

The hexadecimal numbers can contain both upper and lowercase letters a-f and A-F, but when printing them convert them all to uppercase. The possible leading zeroes are also treated as part of the number and should also be printed. Since the numbers are unsigned whole numbers no sign or decimal point is considered as part of the number.

If for example the content of the file input.txt is:

The\_sum\_is

64

and\_the\_minimum\_is

def

ined\_as

a

035

where\_signs\_like

+12AB

are\_not\_considered

as\_well\_as

decimal\_point\_like

12.f5a

also\_not

this1

or

2one

the program should print:

64

DEF

A

035

Total: 4

# 19.

Write a program that will repeat k times each single occurrence of a vowel in the input file "sp.txt" to a new file "output.txt". The first line of the input file contains only the parameter k. The first line (containing the parameter k) should not to be written in the output file.

| **Input** | **Result** |
| --- | --- |
| 3  Do you ever wish you knew exactly what your dog is thinking? A team of designers from the Nordic Society for Invention and Discovery is playing Doctor Dolittle -- they've developed a doggie headset that can read animal brainwaves and translate them into human speech. That's just one of the many amazing scientific breakthroughs we've witnessed this week. | Dooo yooouuu eeeveeer wiiish yooouuu kneeew eeexaaactly whaaat yooouuur dooog iiis thiiinkiiing? AAA teeeaaam ooof deeesiiigneeers frooom theee Nooordiiic Sooociiieeety fooor IIInveeentiiiooon aaand Diiiscoooveeery iiis plaaayiiing Doooctooor Doooliiittleee -- theeey'veee deeeveeelooopeeed aaa doooggiiieee heeeaaadseeet thaaat caaan reeeaaad aaaniiimaaal braaaiiinwaaaveees aaand traaanslaaateee theeem iiintooo huuumaaan speeeeeech. Thaaat's juuust oooneee ooof theee maaany aaamaaaziiing sciiieeentiiifiiic breeeaaakthrooouuughs weee'veee wiiitneeesseeed thiiis weeeeeek. |

# 20.

In given file datnum.txt find and print the row, where the ratio of digits/letters is largest. If there are more than one such rows print the last. No row is longer than 100 characters, and if in a row there are no letters, that row is ignored.

**Example**

Input

aaa123aa 222aa5a289

aaaa cde 23aaaa

45 ,, 56! 7

222aa5a289 aaa123aa

123 aaa 89994 ghj 90 xyz

xyz ,. hgfd

Output

222aa5a289 aaa123aa

**For example:**

| **Input** | **Result** |
| --- | --- |
| dct4MaGaxilBB  EjjcbpTxyq1Np33VgX8zzgYU27kPgNXqh8xGKDv7lon7skaedIdTBGCmqObHiJVJa3Li6qOezrflzrE  lwmxpqkAwVxY7XhKUB4wlQvtrFfwo5X406slhHEryb44K0l9ppaoEvlrh1Vnj0p3yr4oXxCvy  rwfcwTkpzt90nkm3guA1uncNotkbJl1pquSdzuCgzBxbqP4bowyf9T4n0iEsx39aqhl3vh31bBq7asQ  6l1zv000000003333333333TakyQ  dLcGA4yseDppd6s3CUzurl2Xhh6eeX5SpSW7o0DcilYmtSljk5g19lV722jljr | 6l1zv000000003333333333TakyQ |

# 21.

From SI are read dimension and elements of square matrix (MAX 100x100) of integers.

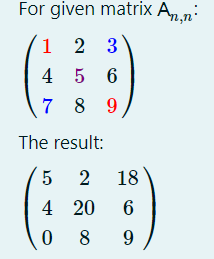
Create a new matrix where each element of the main diagonal will be replaced with the sum of all elements **from the same row right from the element** and elements **from the same column over the element**.

Also replace each element of the antidiagonal with the sum of all elements **from the same row before the element** and elements **from the same column below the element**.

The blue red elements are part of the main diagonal, the blue elements are part of antidiagonal, and the element in purple is part of both diagonals.

If there is no such elements, the element is 0. If some element is on both diagonals, both sums are summed.

**Example:**



**For example:**

| **Input** | **Result** |
| --- | --- |
| 4  1 1 1 1  2 2 2 2  3 3 3 3  4 4 4 4 | 3 1 1 12  2 5 11 2  3 7 6 3  0 4 4 6 |

# 22.

Write a program that will read from SI array of natural numbers (not more than 100). Print all the elements of the array that represent **strong numbers** (each in new line). One number is strong if the sum of factorials of each of his digits is equal to the number itself. Example, number 145 is strong because:

145 = 1! + 4! + 5!

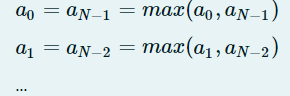
If a number is strong should be checked in a separate function, and for computing factorial a separate recursive function should be implemented and used.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 2  5034  546 | Strong numbers: |

# 23.

Read a an array of N integers from standard input (first read the number N than the elements). Than call a recursive function on the array that will change on the following way:



where *max* is a function that returns the maximum of the two numbers. The function should return the number of switched elements. Then print that number and the changed array.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 5  4 5 6 7 8 | 2  8 7 6 7 8 |

# 24.

Write a fully recursive function triangle (n) which given the input argument n, prints a triangle from the numbers from 1 to n. Additional functions can be used, but no loops should be used.

Example for n=4.

1

12

123

1234

**For example:**

| **Input** | **Result** |
| --- | --- |
| 4 | 1  12  123  1234 |

# 25.

**N** positive integers are entered from the standard input and stored in an array **a (N <100)\*\*. Also, a single-digit integer \*\*k\*\* is entered. Write a function that will find the maximum of the abbreviated numbers of the elements in the array \*\*a\*\* and provide a call to that function in the main () function. A given element of the array is abbreviated as follows: all digits smaller than or equal to \*\*k** are deleted from the number.

For example: If k = 3 and a[0] = 12345, then abbreviated a[0] becomes 45, if a[1] = 458763, a[1] becomes 45876, and a[2] = 112 is shortened to a[2] = 0. The maximum abbreviated number is 45876.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 3  12365  98765  654  2 | 98765 |

# 26.

**N** positive integers are entered from the standard input and are stored in an array **(N <100)**. All elements that meet the next criteria should be presented on the standard output in the same order that they have appeared on the standard input. The criteria is following: the most important digit (the digit on the first place) is odd and the least important digit is even. If, no such number was entered, you should print out "No such elements!" message.

The implementation of checking the criteria should be performed in a separate function (recursive function for additional 5 points).

**For example:**

| **Input** | **Result** |
| --- | --- |
| 10  12345  2345  345678  2912301  3212342  9129321  77777  213312  32322  313138 | 345678  3212342  32322  313138 |

# 27.

Read the dimension N (with a maximum value of 100) and the values (possible values are only 1 and 0) of one square matrix $A\_ {NxN}$ from the standard input. Create a new matrix $B\_ {NxN}$ in the following way:

For each element in B, write the value of the same corresponding element from A. During this procedure, if in the neighborhood of one element of A (neighbors are the elements above it or below it, on the left or right side of it) there are minimum 3 elements with a value of 1, then that element in the matrix B gets a value of 1 (although in A the corresponding value was zero).

Print the newly obtained matrix B.

Example:

Matrix А:

1 0 0 1 0

1 1 0 1 0

1 1 1 1 1

0 1 0 0 1

1 0 1 1 0

Matrix B:

1 0 0 1 0

1 1 1 1 0

1 1 1 1 1

1 1 1 1 1

1 1 1 1 0

**For example:**

| **Input** | **Result** |
| --- | --- |
| 10  1 1 0 0 0 0 0 0 1 1  0 1 1 1 0 0 1 1 1 1  0 0 1 0 0 1 0 1 0 0  0 1 1 0 1 0 0 0 1 0  1 0 0 0 0 1 0 1 0 1  0 1 0 1 0 0 0 0 1 0  0 0 0 1 0 1 0 0 1 1  1 0 0 1 1 1 0 0 0 1  1 1 1 1 0 1 0 0 0 0  0 0 0 1 1 0 1 1 0 1 | 1 1 0 0 0 0 0 0 1 1  0 1 1 1 0 0 1 1 1 1  0 1 1 0 0 1 1 1 1 0  0 1 1 0 1 1 0 1 1 0  1 1 0 0 0 1 0 1 1 1  0 1 0 1 0 0 0 0 1 1  0 0 0 1 1 1 0 0 1 1  1 0 0 1 1 1 0 0 0 1  1 1 1 1 1 1 0 0 0 0  0 0 0 1 1 1 1 1 0 1 |

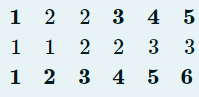
# 28.

Firstly read the dimension of one matrix (m, n <= 100) from the standard input, and then the elements of the matrix. Generate an array (with at most m x n elements) of the elements of the matrix that will contain the elements of each row that appear exactly once in that row. The order of writing the elements in the array corresponds to the order of the rows and the order of the elements within the row. If the newly generated array has no elements, print N, otherwise, print the elements of the array.

Example:

*Input:*

3 6



Output:\_

1 3 4 5 1 2 3 4 5 6

**For example:**

| **Input** | **Result** |
| --- | --- |
| 7  5  30  -10  45  -10  12  23  23  56  78  23  -12  -12  32  99  0  -1  -100  33  33  12  -7  -4  -7  -4  -23  -12  78  34  23  98  654  43  56  67  89 | 30 45 12 56 78 32 99 0 -1 -100 12 -23 -12 78 34 23 98 654 43 56 67 89 |

# 29.

Read the dimensions M and N (not bigger than 100) and the values (whole numbers) of one matrix **A\_ {MxN}** from the standard input. Print out the index of the first column that has the smallest sum of all of its elements.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 3 8  1 2 3 5 7 11 13 17  19 23 29 31 37 41 43 47  53 59 61 67 71 73 79 83 | 0 |

# 30.

A text file "input.txt" consist of digits and letters. The number of digits and letters in a row is limited to 80. Write a program that for every row will print out the letters concatenated next to each other first followed by the sum of all numbers that appear in that row. If there are no digits in the row, for the sum of the numbers you should print out 0.

Example:  For the text file:

y2d3vbqiiK064waaa

ksdjdjgdjgHH

the output is following:

ydvbqiiKwaaa69

ksdjdjgdjgHH0

**For example:**

| **Input** | **Result** |
| --- | --- |
| 7h3r3545h4d0wJu57B3h1ndM3  5hr0ud1n63v3ryBr347h174k3  # | hrhdwJuBhndM3623  hrudnvryBrhk596 |

# 31.

The file input.txt contains a single word that is not longer than 30 letters in each row. Find all the words in a file that define a palindrome only with the vowels they contain (after making the opposite word of the given one, the vowels have not changed their indexes within the word). Print this words on a standard output. The appearance of uppercase and lowercase letters should be ignored.

Example:

input.txt

Hellen

input

werAnifameb

output:

Hellen

werAnifameb

**For example:**

| **Input** | **Result** |
| --- | --- |
| KalAbaLAm  BananaS  WooooSH  ANMA  # | KalAbaLAm  BananaS  ANMA |

# 32.

A text file "rectangle.in" is given. In the first row of the file there is an integer N, and in the second row, an array of N single-digit positive numbers. Find the numbers A and B that repeat the most times in the array. Then, print the rectangle with dimension A x B on the standard output, in the following format:

\*\*.....\*

\*\*.....\*

\*\*.....\*

\*\*.....\*

\*\*.....\*

(A lines with B-stars, where A is the smaller of the two numbers found).

In the test cases there are always exactly two numbers that appear the same most times.

Example:

Input:

10

2 5 3 7 5 3 3 6 4 5

(3 appears 3 times, 5 appears 3 times, while the others are less frequent)

Output:

\* \* \* \* \*

\* \* \* \* \*

\* \* \* \* \*

**For example:**

| **Input** | **Result** |
| --- | --- |
| 11  9 8 9 8 1 1 9 8 7 6 5 | \* \* \* \* \* \* \* \* \*  \* \* \* \* \* \* \* \* \*  \* \* \* \* \* \* \* \* \*  \* \* \* \* \* \* \* \* \*  \* \* \* \* \* \* \* \* \*  \* \* \* \* \* \* \* \* \*  \* \* \* \* \* \* \* \* \*  \* \* \* \* \* \* \* \* \* |

# 33.

Write a program that will print on SO the contents of given textual file "text.txt". While printing, in front of each row should print the ratio of upper/lower case letters in the row. Maximal length of the row is 80 characters. At the end print the number of the row with largest ratio of upper/lower case letters.

**For example:**

| **Input** | **Result** |
| --- | --- |
| So, so you think you can tell Heaven from Hell,  blue skies From Pain.  Can you tell a Green Field From a cold steel rail?  A smile from a veil?  Do you think you can tell?  And did they geT you to Trade Your heroes for ghosts?  Hot ashes for trees? | 0.09 So, so you think you can tell Heaven from Hell,  0.13 blue skies From Pain.  0.11 Can you tell a Green Field From a cold steel rail?  0.07 A smile from a veil?  0.05 Do you think you can tell?  0.11 And did they geT you to Trade Your heroes for ghosts?  0.07 Hot ashes for trees?  1 |

# 34.

The file *\* input.txt \** is given and contains a single word that is not longer than 20 letters in each line. Write a program that will print the word that has the biggest number of different letters. Words that have less than four letters are not taken into account during the check. Do not make a difference between upper and lower case letters. If there are more words that meet the criteria, the last one is printed.

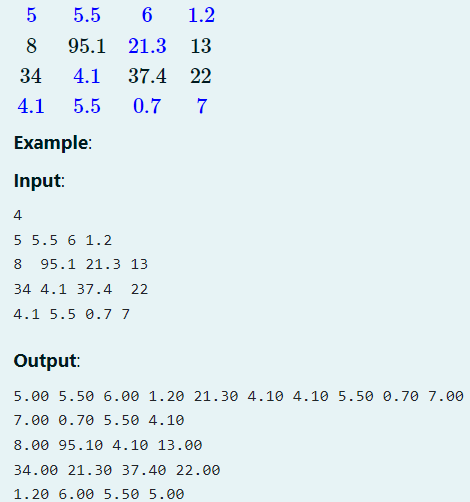
**For example:**

| **Input** | **Result** |
| --- | --- |
| To  those  whom  Ive  fought  with  and  to  those  I  dont  know  your  name  we  fought  by  one  another  You  did  not  die  in  vain  # | another |

# 35.

**Z-diagonal** of given square matrix is composed of the elements of the first row, the elements of the supportive diagonal, and the elements of the last row (marked with blue color on the example).

From SI read the dimensions of square matrix (integer N, N > 2, N<100), and the elements of the matrix (real numbers). Store the elements of the Z-diagonal in a array, starting from the first row, continuing with the supportive diagonal, and finishing with the last row (the last element of the first row is the same element with the first element of the supportive diagonal, and the last element of the supportive diagonal is the same as the first element of the last row, and these elements should be stored only **once**). The resulted array should be printed on the SO. Then print the original matrix, but with the elements of Z-diagonal in reverse order.



**For example:**

| **Input** | **Result** |
| --- | --- |
| 4  5 5.5 6 1.2  8 95.1 21.3 13  34 4.1 37.4 22  4.1 5.5 0.7 7 | 5.00 5.50 6.00 1.20 21.30 4.10 4.10 5.50 0.70 7.00  7.00 0.70 5.50 4.10  8.00 95.10 4.10 13.00  34.00 21.30 37.40 22.00  1.20 6.00 5.50 5.00 |

# 36.

class

A whole number x ( 1<=x <=10) is read from SI.

After that, an array of no more than 100 integers is read. First the count n of numbers to follow is given, and then the n numbers.

Write a program that will print in descending order the numbers from the array that contain exactly x unique digits.

The calculation of the count of unique digits for a given number should be performed in a separate function. Solutions without functions will be graded with a maximum of 40% of the points.

Example: Input:

4 10 99 190 10100 5931 1232 999 1231 9999 1234567 2342421 Output:

2342421 5931

Explanation: from the read numbers, only 5931 and 2342421 have 4 unique digits. These two numbers are printed in descending order.

# 37.

A text file named “input.txt” is given. The lines in the file contain no more than 100 characters. Write a program that for each line of the file will print information for the number of letters, number of words, and the line itself. Words are considered all combinations of letters and digits separated with space or punctuation marks. The printing format is the following:

b, w: row

Where b is the count of letters and digits, w is the count of words and row is the content of the line.

Example:

// file input.txt You have been in the pipeline Filling in time Provided with toys and ‘scouting for boys’ You brought a guitar to punish your ma And you didn’t like school And you know you are nobody fool So welcome to the machine

Output: 24, 6: You have been in the pipeline 13, 3: Filling in time 34, 7: Provided with toys and ‘scouting for boys’ 31, 8: You brought a guitar to punish your ma 22, 6: And you didn’t like school 26, 7: And you know you are nobody fool 21, 5: So welcome to the machine

# 38.

The file matrica.txt contains the elements of an integer matrix (A). The number of rows n and the number of columns m of the matrix (n, m < 120) are written in the first line of the file. After them, in each line of the file, the elements of each row of the matrix are written.

Print the number of index numbers in each column.

An element in a matrix is considered for index number if its value is identical with the value that is obtained by concatenation of the index of the row and the index of the column of the element.

Help:

Index element: The value of the element Aij is the same with ij as a number.

* If i=10, j=31, the element Aij should have the value of 1031 in order to be an index number.

Example:

Input: // file matrica.txt

4 5

0 1 2 3 4

5 6 7 8 9

10 21 22 23 11

0 1 2 33 4

Output:

1

2

2

3

1

Explanation:

In the matrix A , index elements are:

0 column: 0

1st column: 1, 21

2nd column: 2, 22

3rd column: 3, 23, 33

4th column: 4

# 39.

Strings recording

A number N (N<100) and then N strings are read from SI. The strings characters contain letters, digits and special characters and each of them contains no more than 50 characters.

Write a program that will print on the screen all the strings in which the substring A1c (A1C, a1C, a1c the case doesn’t matter) appears at least 2 times. The printing should be done in the same order in which the strings were read. Also, when printing the strings, all letters should be printed in lowercase.

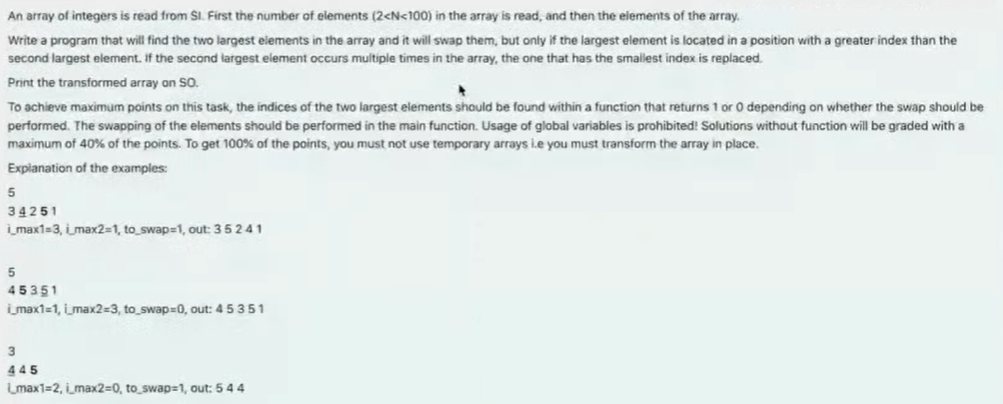
stringssss

# 40.

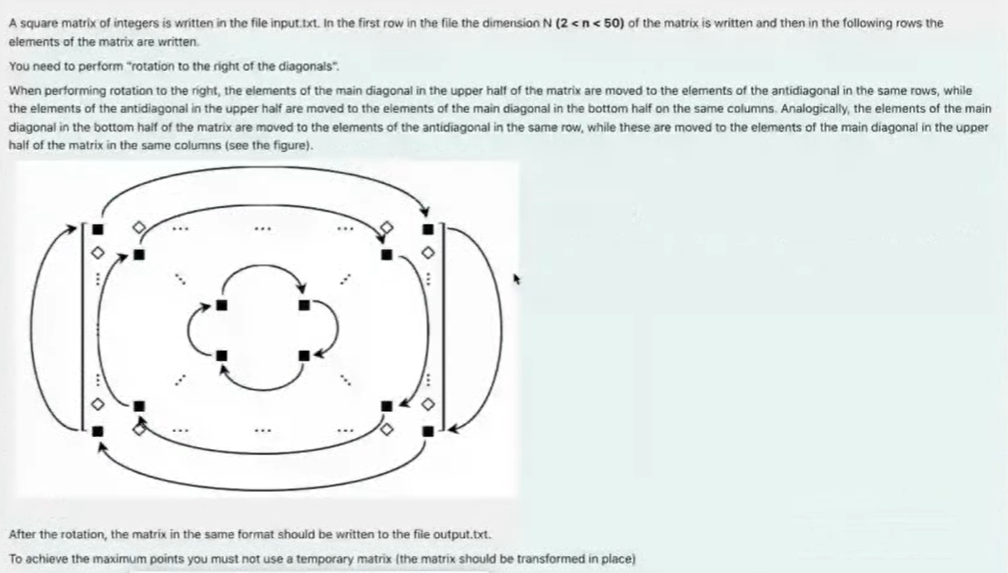
Exams recording



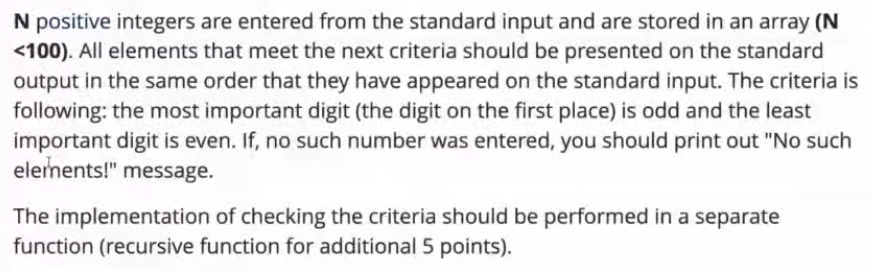
# 41.



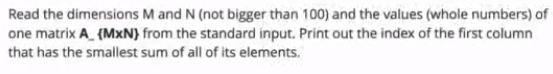
# 42.



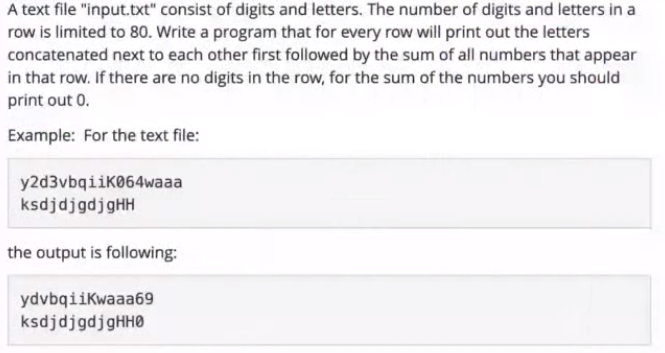
# 43.



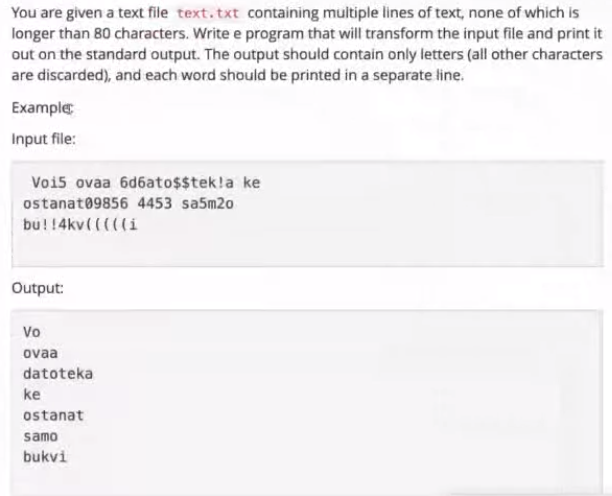
# 44.



# 45.



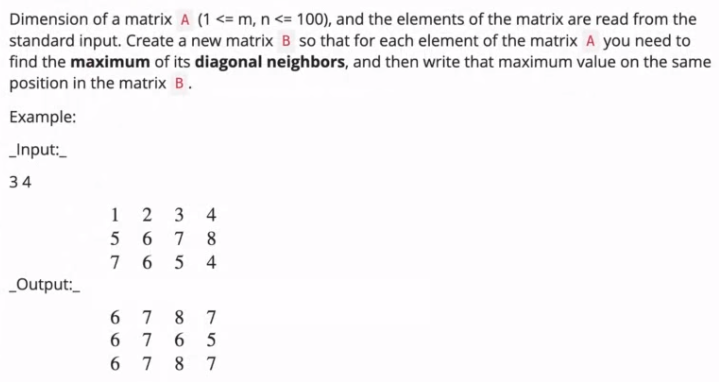
# 46.



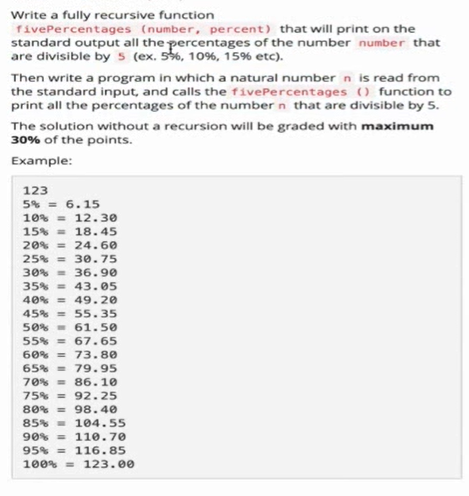
Rec exams

# 47.

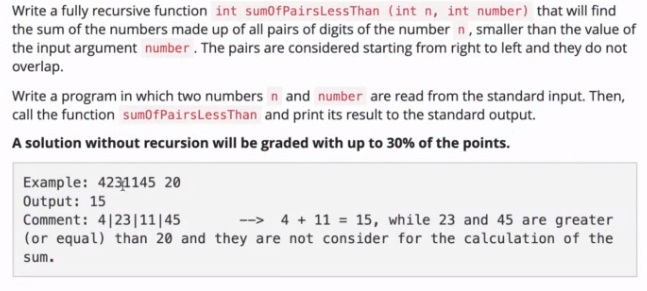
Files



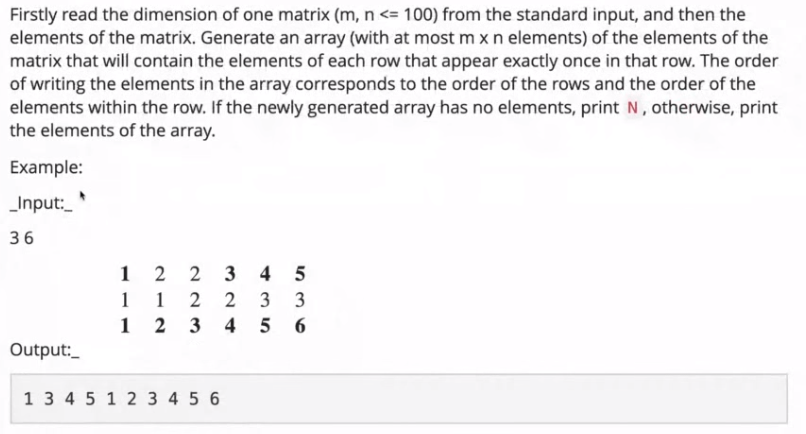
# 48.



# 49.



# 50.



# 51.

SOLUTIONS

# 1.

#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>  
  
int** sumOfHiddenIntegers(**char** \*str){  
 **int** currentNumber=0, sum=0;  
 **for**(**int** i=0 ; i< strlen(str) ; i++){  
 **if**(isdigit(str[i])){  
 currentNumber=currentNumber\*10+(str[i]-**'0'**);  
 } **else**{  
 sum+=currentNumber;  
 currentNumber=0;  
 }  
 }  
 **return** sum;  
}  
  
**int** main(){  
 **char** c;  
 **char** str[100];  
 **int** k=0;  
 **while** (scanf(**"%c"**,&c)){  
 **if**(c==**'!'**){  
 str[k]=**'!'**;  
 str[k+1]=**'\0'**;  
 **break**;  
 }  
 str[k]=c;  
 k++;  
 }  
 printf(**"%d"**, sumOfHiddenIntegers(str));  
 **return** 0;  
}

# 2.

*/\*From SI read N strings not longer than 80 chars. At the beginning of the program read two integers:  
• N - number of strings you are going to read  
• X - shifting size.  
Each of the read strings should be transformed in such a way that lowercase and uppercase letters (a-z, A-Z) are replaced with  
the same letter shifted X places afterwards in the alphabet (a-z). If you overflow the range of alphabet, continue cyclic from  
the beginning of the alphabet. Transformed string should be printed on SO.  
Transformation of a string should be implemented in a separate recursive function.  
Example:  
Welcome -> transformed with shift 5 -> Bjqhtrj  
\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>***//c='a'+(X-('z'-c))***void** transform(**char** \*str,**int** X){  
 **for**(**int** i=0 ; i< strlen(str) ; i++){  
 **if**(islower(str[i])){  
 **if**(str[i]+X>**'z'**){  
 **int** s=**'z'**-str[i];  
 **int** shift=X-s;  
 str[i]=**'a'**-1+shift;  
 } **else**{  
 str[i]=str[i]+X;  
 }  
 } **else if**(isupper(str[i])){  
 **if**(str[i]+X>**'Z'**){  
 **int** s=**'Z'**-str[i];  
 **int** shift=X-s;  
 str[i]=**'A'**-1+shift;  
 } **else**{  
 str[i]=str[i]+X;  
 }  
 }  
 }  
}  
  
**void** transformRecursive(**char** \*str, **int** X){  
 **if**(str==**NULL**){  
 **return**;  
 } **else**{  
 **if**(islower(\*str)){  
 **if**(\*str+X>**'z'**){  
 **int** s=**'z'**-\*str;  
 **int** shift=X-s;  
 \*str=**'a'**-1+shift;  
 **return** transformRecursive(str+1,X);  
 } **else**{  
 \*str=\*str+X;  
 **return** transformRecursive(str+1,X);  
 }  
 } **else if**(isupper(\*str)){  
 **if**(\*str+X>**'Z'**){  
 **int** s=**'Z'**-\*str;  
 **int** shift=X-s;  
 \*str=**'A'**-1+shift;  
 **return** transformRecursive(str+1,X);  
 } **else**{  
 \*str=\*str+X;  
 **return** transformRecursive(str+1,X);  
 }  
 }  
 }  
}  
  
**int** main(){  
 **int** N,X;  
 scanf(**"%d%d\n"**,&N,&X);  
 **char** str[80];  
 **for**(**int** i=0 ; i< strlen(str) ; i++){  
 fgets(str,80,**stdin**);  
 str[strlen(str)-1]=**'\0'**;  
 transformRecursive(str,X);  
 puts(str);  
 }  
 **return** 0;  
}

# 3.

*/\*Write a program that will print the contents of the text file sifra.txt coded.  
The coding is done letter by letter, so that each letter (only letters, not digits, dots, ...)  
is replaced with the letter 3 places after in the alphabet. In case of the letters at the end of  
the alphabet, the counting continues cyclic from the beginning of the alphabet. In the final coded  
text all letters are lowercase. Example letter a is coded to d, H is coded to k, p is coded to s, x is coded to a, etc.\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>***//c='a'+(X-('z'-c))***void** transform(**char** c){  
 **if**(isalpha(c)){  
 c=tolower(c);  
  
 **if**(c+3>**'z'**){  
 **int** s=**'z'**-c;  
 **int** shift=3-s;  
 c=**'a'**-1+shift;  
 } **else**{  
 c=c+3;  
 }  
 }  
 printf(**"%c"**,c);  
}  
  
**int** main(){  
 FILE \*f= fopen(**"C:\\Users\\user\\Desktop\\CCC\\Exams\\sifra.txt"**,**"r"**);  
 **char** c;  
 **while**((c= fgetc(f))!=**EOF**){  
 transform(c);  
 }  
 fclose(f);  
 **return** 0;  
}

# 4.

*/\*Let A = {a1, a2, .. a\_n} is an array of integers. The size of the array n and the elements  
of the array are read from SI. Write a program that will transform the array so each element  
of the original array is replaced with the number of appearance of the least significant digit  
(the right most) in the number itself. Print the result array on the standard output.  
Compute the count of given digit in a number with separate recursive function.  
Example:  
Input:  
5  
1 11 1121 111222112 22222  
Output:  
1 2 3 4 5  
\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>  
  
int** countOfOcc(**int** number,**int** ld){  
 **if**(number==0){  
 **return** 0;  
 } **else**{  
 **if**(ld==number%10){  
 **return** 1+ countOfOcc(number/10,ld);  
 } **else**{  
 **return** countOfOcc(number/10,ld);  
 }  
 }  
}  
  
**int** main(){  
 **int** n;  
 scanf(**"%d"**,&n);  
 **int** A[100];  
 **for**(**int** i=0 ; i<n ; i++){  
 scanf(**"%d"**,&A[i]);  
 }  
 **for**(**int** i=0 ; i<n ; i++){  
 **int** lastDigit=A[i]%10;  
 A[i]= countOfOcc(A[i],lastDigit);  
 }  
 **for**(**int** i=0 ; i<n ; i++){  
 printf(**"%d "**,A[i]);  
 }  
 **return** 0;  
}

# 5.

*/\*Write a program that will read unknown number of three integers, and for each triple (a, b, c)  
will print the number (a or b) that contains the digit c more times on even positions  
(the positions are counted from right to left, and the first position is 1).\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>  
  
int** countOccOnEvenPos(**int** number, **int** c){  
 **int** counter=0;  
 number=number/10;  
 **while**(number){  
 **int** ld=number%10;  
 **if**(ld==c){  
 counter++;  
 }  
 number/=100;  
 }  
 **return** counter;  
}  
  
**int** main(){  
 **int** a,b,c;  
 **while**(scanf(**"%d%d%d"**,&a,&b,&c)){  
 **int** countA= countOccOnEvenPos(a,c);  
 **int** countB= countOccOnEvenPos(b,c);  
 **if**(countA>countB){  
 printf(**"%d\n"**,a);  
 } **else if**(countB>countA){  
 printf(**"%d\n"**,b);  
 }  
 }  
 **return** 0;  
}

# 6.

*/\*In a given file "numbers.txt" are written more rows with integers, and each row  
starts with one integer (N >= 1) that represents how many integers are following  
afterwards in the same row. Write a program that on SO will print print the number  
with largest most significant digit. Reading of integers ends when 0 is read.\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>**#include **<math.h>  
  
int** countOfDigits(**int** number){  
 **int** counter=0;  
 **while**(number){  
 counter++;  
 number/=10;  
 }  
 **return** counter;  
}  
  
**int** main(){  
 FILE \*f= fopen(**"C:\\Users\\user\\Desktop\\CCC\\Exams\\numbers.txt"**,**"r"**);  
 **int** n;  
 **int** A[100];  
 **for**(**int** i=0 ;; i++){  
 fscanf(f,**"%d"**,&n);  
 **if**(n==0){  
 **break**;  
 }  
 **for**(**int** j=0 ; j<n ; j++){  
 fscanf(f,**"%d"**,&A[j]);  
 }  
 **int** d= countOfDigits(A[0]);  
 **int** significantDig=A[0]/(**int**)pow(10,d-1);  
 **int** max=significantDig;  
 **int** numMax=A[0];  
 **for**(**int** j=0 ; j<n ; j++){  
 d= countOfDigits(A[j]);  
 significantDig=A[j]/(**int**)pow(10,d-1);  
 **if**(significantDig>max){  
 max=significantDig;  
 numMax=A[j];  
 }  
 }  
 printf(**"%d\n"**,numMax);  
 }  
  
 fclose(f);  
 **return** 0;  
}

# 7.

*/\*Write a program that reads matrix with dimensions MxN (max. 100x100). At the beginning  
 the dimensions are read, and then the elements of the matrix with values 0 or 1.  
Your program should count and print on SO the number of rows and columns with at least 3 consecutive  
appearance of elements with value 1.\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>**#include **<math.h>  
  
int** main(){  
 **int** m,n;  
 scanf(**"%d%d"**,&m,&n);  
 **int** matrix[100][100];  
 **for**(**int** i=0 ; i<m ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 scanf(**"%d"**,&matrix[i][j]);  
 }  
 }  
 **int** countRows=0;  
 **int** countColumns=0;  
 **for**(**int** i=0 ; i<m ; i++){  
 **for**(**int** j=0 ; j<n-2 ; j++){  
 **if**(matrix[i][j]==1 && matrix[i][j+1]==1 && matrix[i][j+2]==1){  
 countRows++;  
 }  
 }  
 }  
  
 **for**(**int** j=0 ; j<n ; j++){  
 **for**(**int** i=0 ; i<m-2 ; i++){  
 **if**(matrix[i][j]==1 && matrix[i+1][j]==1 && matrix[i+2][j]==1){  
 countColumns++;  
 }  
 }  
 }  
 printf(**"%d"**,countColumns+countRows);  
 **return** 0;  
}

# 8.

*/\*Read from standard input the dimensions of a matrix (m < 100 and n < 100) which contains only zeros and ones.  
Then read the elements of the matrix.  
Read the coordinates of one element of the matrix. If the element of the read position is 0, then it should become 1 and:  
• all zeros above that element to the first element with value 1 or to the start of the column,  
• all zeros bellow that element to the first element with value 1 or to the end of the column,  
• all zeros left from that element to the first element with value 1 or to the begin of the row and  
• all zeros right from that element to the first element with value 1 or to the end of the row.  
If the element of the given position is 1 than the matrix stays unchanged.  
Print the changed matrix.\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>**#include **<math.h>  
  
int** main(){  
 **int** m,n;  
 scanf(**"%d%d"**,&m,&n);  
 **int** matrix[100][100];  
  
 **for**(**int** i=0 ; i<m ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 scanf(**"%d"**,&matrix[i][j]);  
 }  
 }  
 **int** row,column;  
 scanf(**"%d%d"**,&row,&column);  
  
 **if**(matrix[row][column]==0){  
 **for**(**int** i=0 ; i < m ; i++){  
 matrix[row][column]=1;  
 **if**(matrix[row - i - 1][column] == 0){  
 matrix[row - i - 1][column]=1;  
 } **else**{  
 **break**;  
 }  
 **if**(matrix[row + i + 1][column] == 0){  
 matrix[row + i + 1][column]=1;  
 }  
 }  
 **for**(**int** i=0 ; i < m ; i++){  
 **if**(matrix[row + i + 1][column] == 0){  
 matrix[row + i + 1][column]=1;  
 } **else**{  
 **break**;  
 }  
 }  
 **for**(**int** j=0 ; j < n ; j++){  
 **if**(matrix[row][column - j - 1] == 0){  
 matrix[row][column - j - 1]=1;  
 } **else**{  
 **break**;  
 }  
 }  
 **for**(**int** j=0 ; j < n ; j++){  
 **if**(matrix[row][column+ j + 1] == 0){  
 matrix[row][column+ j + 1]=1;  
 } **else**{  
 **break**;  
 }  
 }  
  
 }  
  
 **for**(**int** i=0 ; i<m ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 printf(**"%d "**,matrix[i][j]);  
 }  
 printf(**"\n"**);  
 }  
 **return** 0;  
}

# 9.

*/\*The dimensions of one matrix (m,n <= 100), and then the elements of the matrix are entered from the keyboard.  
Make a transformation of the matrix so that the middle element in each column is replaced by the difference  
(by absolute value) of the sum of the elements in the first half of the column and the sum of the elements in  
the second half of the column. If the matrix has even number of rows, the value of the middle two elements should  
be changed. The middle element(s) are included when calculating the sum (in the case of an odd number of rows,  
the middle element is included when calculating both of the sums (upper and lower sum)!).  
Print the transformed matrix on the standard output.\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>**#include **<math.h>  
  
int** main(){  
 **int** m,n;  
 scanf(**"%d%d"**,&m,&n);  
 **int** matrix[100][100];  
 **for**(**int** i=0 ; i<m ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 scanf(**"%d"**,&matrix[i][j]);  
 }  
 }  
  
  
 **if**(m%2==0){  
 **for**(**int** j=0 ; j<n ; j++){  
 **int** k=0;  
 **int** lowerSum=0,upperSum=0;  
 **int** sum=0;  
 **while**(k<m/2){  
 upperSum+= matrix[k][j];  
 k++;  
 }  
 **while**(k>=m/2 && k<m){  
 lowerSum+=matrix[k][j];  
 k++;  
 }  
 sum=upperSum-lowerSum;  
 **if**(sum<0){  
 matrix[m/2][j]= -1\*sum;  
 matrix[(m/2)-1][j]= -1\*sum;  
 } **else**{  
 matrix[m/2][j]= sum;  
 matrix[(m/2)-1][j]= sum;  
 }  
 }  
 } **else**{  
 **for**(**int** j=0 ; j<n ; j++){  
 **int** k=0;  
 **int** lowerSum=0,upperSum=0;  
 **int** sum=0;  
 **while**(k<=m/2){  
 upperSum+= matrix[k][j];  
 k++;  
 }  
 k=m/2;  
 **while**(k>=m/2 && k<m){  
 lowerSum+=matrix[k][j];  
 k++;  
 }  
 sum=upperSum-lowerSum;  
 **if**(sum<0){  
 matrix[m/2][j]= -1\*sum;  
 } **else**{  
 matrix[m/2][j]= sum;  
 }  
 }  
 }  
  
  
 **for**(**int** i=0 ; i<m ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 printf(**"%d "**,matrix[i][j]);  
 }  
 printf(**"\n"**);  
 }  
 **return** 0;  
}

# 10.

*/\*Write a recursive function that will find the number of odd numbers of an array of integers.  
The function accepts as an arguments the array for which the number of odd numbers is required  
nd the total number of elements in the array. The function is given with the following prototype:  
int Odd (int a [], int n);  
Also write the main () function to test the Odd function.  
\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>**#include **<math.h>  
  
int** Odd(**int** a[], **int** n){  
 **if**(n==0){  
 **if**(a[0]%2!=0){  
 **return** 1;  
 } **else**{  
 **return** 0;  
 }  
 } **else**{  
 **if**(a[n]%2!=0){  
 **return** 1+ Odd(a,n-1);  
 } **else**{  
 **return** Odd(a,n-1);  
 }  
 }  
}  
  
**int** main(){  
 **int** n;  
 scanf(**"%d"**,&n);  
 **int** a[100];  
 **for**(**int** i=0 ; i<n ; i++){  
 scanf(**"%d"**,&a[i]);  
 }  
 **int** odd= Odd(a,n);  
 printf(**"%d"**,odd);  
 **return** 0;  
}

# 11.

*/\*For one integer b we say that is a parity complement of other integer a if and only if in the number b  
all the even digits of the number a are replaced with the corresponding (odd) digit increased by 1,  
and all odd digits of the number a are replaced with the corresponding (odd) digit decreased by 1 .  
Example. Number b = 4013698 is a parity complement of a = 5102789.  
Read from SI unknown number of integers (not more than 100), until something that is not a number is entered.  
Your task is to print the smallest 5 of the parity complements of all numbers, from the smallest to the largest.  
Notice: If less then 5 numbers are entered, then print all the entered numbers.  
Finding the parity complement of a number should be implemented in a separate recursive function parcom(int a).  
Example.  
For numbers: 9641, 69403, 103007, 6, 11, 181, 20486 you should find their parity complements  
(those are: 8750, 78512, 12116, 7, 0, 90 and 31597, respectively), and print the 5 smallest of  
them in this order: 0 7 90 8750 12116.  
Using global variables is FORBIDDEN.  
\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>**#include **<math.h>  
  
int** countOfDigits(**int** number){  
 **int** countOfDig=1;  
 **while**(number){  
 number/=10;  
 countOfDig\*=10;  
 }  
 countOfDig/=10;  
 **return** countOfDig;  
}  
*//int restN=a%(int) pow(10,d-1);  
//a%(int) pow(10,d-1)  
  
//int d= countOfDigits(a);  
// int fd=a%d;***int** parcom(**int** a){  
 **int** d= countOfDigits(a);  
 **int** c= countOfDigits(a);  
 **int** parcom=0;  
 **while** (d){  
 **int** fd=a/d;  
 **if**(fd%2==0){  
 parcom=10\*parcom+(fd+1);  
 } **else**{  
 parcom=10\*parcom+(fd-1);  
 }  
 a=a%d;  
 d/=10;  
 }  
 **return** parcom;  
}  
**void** bubbleSort(**int** a[],**int** size){  
 **for**(**int** i= 0 ; i<size-1 ; i++){  
 **for**(**int** j= 0 ; j<size-i-1 ; j++){  
 **if**(a[j]>a[j+1]){  
 **int** tmp=a[j];  
 a[j]=a[j+1];  
 a[j+1]=tmp;  
 }  
 }  
 }  
}  
  
**int** main(){  
 **int** a;  
 **int** array[100];  
 **int** i=0;  
 **while**(scanf(**"%d"**,&a)){  
 **int** pc= parcom(a);  
 array[i]= pc;  
 i++;  
 }  
 bubbleSort(array,i);  
 **for**(**int** j=0 ; j < 5 ; j++){  
 printf(**"%d "**,array[j]);  
 }  
 **return** 0;  
}

# 12.

*/\*Write a program where a matrix A read from SI (maximum value of dimensions of the matrix is 100)  
is transformed to a new matrix B. Each element of the new matrix B is sum of positive neighbors of  
 the corresponding element of the matrix A. Print the new matrix B on the standard output.\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>**#include **<math.h>  
  
  
int** main(){  
 **int** m,n;  
 scanf(**"%d%d"**,&m,&n);  
  
 **int** A[100][100];  
  
 **for**(**int** i=0 ; i<m ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 scanf(**"%d"**,&A[i][j]);  
 }  
 }  
 **int** B[100][100];  
 **for**(**int** i=0 ; i<m ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 **int** sum=0;  
 **if**(A[i][j-1]>=0){  
 sum+=A[i][j-1];  
 }  
 **if**(A[i][j+1]>=0){  
 sum+=A[i][j+1];  
 }  
 **if**(A[i-1][j]>=0){  
 sum+=A[i-1][j];  
 }  
 **if**(A[i+1][j]>=0){  
 sum+=A[i+1][j];  
 }  
 B[i][j]=sum;  
 **if**(A[i][j-1]<0 && A[i][j+1]<0 && A[i+1][j]<0 && A[i-1][j]<0){  
 B[i][j]=A[i][j];  
 }  
 }  
 }  
  
 **for**(**int** i=0 ; i<m ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 printf(**"%d "**,B[i][j]);  
 }  
 printf(**"\n"**);  
 }  
  
 **return** 0;  
}

# 13.

*/\*Write a program that will read a string (not longer than 100 characters) containing letters and digits.  
The program should print on the standard output the letters that are between the first and the second digit,  
left-to-right, including the bordering digits themselves. If there is only one digit in the string, the program  
should print all the letters till the end of the string. If there is no digits in the string the program should  
print out the message "No digits".\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>**#include **<math.h>  
  
int** countD(**char** \*str){  
 **int** counter=0;  
 **for**(**int** i=0 ; i< strlen(str) ; i++){  
 **if**(isdigit(str[i])){  
 counter++;  
 }  
 }  
 **return** counter>0;  
}  
  
**int** main(){  
 **char** str[100];  
 fgets(str, 100,**stdin**);  
 str[strlen(str)-1]=**'\0'**;  
 **if**(countD(str)){  
 **int** i;  
 **for**(i=0 ; i< strlen(str) ; i++){  
 **if**(isdigit(str[i])){  
 **break**;  
 }  
 }  
 **int** j;  
 **for**(j=i+1 ; j< strlen(str) ; j++){  
 **if**(isdigit(str[j])){  
 **break**;  
 }  
 }  
 str[j+1]=**'\0'**;  
 puts(str+i);  
 } **else**{  
 printf(**"No digits"**);  
 }  
 **return** 0;  
}

# 14.

*/\*An array of integers is to be read from SI. First the number N (N<=100) is entered followed by N integers.  
Write a program that will transform the array in the following way: starting at the beginning of the array,  
for each element that is equal to the next element in the array, double the value of the first one and overwrite  
the second one with 0. After that, rearrange the array so that all the zeroes will be moved at the end of the array,  
keeping the order of the other elements. Print all the elements of the array on the SO in a single line, separated by space.  
Write separate functions for transforming the array and for reordering the elements.  
\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>**#include **<math.h>  
  
void** transform(**int** array[],**int** n){  
 **for**(**int** i=0 ; i<n ; i++){  
 **if**(array[i]==array[i+1]){  
 array[i]=2\*array[i];  
 array[i+1]=0;  
 }  
 }  
 **int** zero[100],countZ=0,nonZero[100],countNZ=0;  
  
 **for**(**int** i=0 ; i<n ; i++) {  
 **if** (array[i] != 0) {  
 nonZero[countNZ] = array[i];  
 countNZ++;  
 } **else** {  
 zero[countZ] = array[i];  
 countZ++;  
 }  
 }  
 **for**(**int** i=0 ; i < countNZ ; i++){  
 array[i]=nonZero[i];  
 }  
 **for**(**int** i=0,j=countNZ ; i < countZ && countNZ < n ; i++,j++){  
 array[j]=zero[i];  
 }  
}  
  
**int** main(){  
 **int** n;  
 scanf(**"%d"**,&n);  
 **int** array[100];  
 **for**(**int** i=0 ; i<n ; i++){  
 scanf(**"%d"**,&array[i]);  
 }  
 transform(array,n);  
 **for**(**int** i=0 ; i<n ; i++){  
 printf(**"%d "**,array[i]);  
 }  
 **return** 0;  
}

# 15.

*/\*Read a square matrix of integers from standard input. First the dimension N is entered (2 < N <= 100)  
and then the elements of the matrix by rows.  
Transform the matrix in a way that all the elements ABOVE the main diagonal will be set to the value of  
the maximal element found above the main diagonal, while all the elements BELOW the main diagonal will be  
set to the value of the minimal element found below the main diagonal. Print out the transformed matrix.\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>**#include **<math.h>  
  
int** main() {  
 **int** n;  
 scanf(**"%d"**,&n);  
 **int** matrix[100][100];  
 **for**(**int** i=0 ; i<n ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 scanf(**"%d"**,&matrix[i][j]);  
 }  
 }  
 **int** max=matrix[0][1];  
 **int** min=matrix[1][0];  
 **for**(**int** i=0 ; i<n ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 **if**(i!=j && i+j>i+i){  
 **if**(matrix[i][j]>max){  
 max=matrix[i][j];  
 }  
 }  
 **if**(i!=j && i+j<i+i){  
 **if**(matrix[i][j]<min){  
 min=matrix[i][j];  
 }  
 }  
 }  
 }  
 **for**(**int** i=0 ; i<n ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 **if**(i!=j && i+j>i+i){  
 matrix[i][j]=max;  
 }  
 **if**(i!=j && i+j<i+i){  
 matrix[i][j]=min;  
 }  
 }  
 }  
 **for**(**int** i=0 ; i<n ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 printf(**"%d "**,matrix[i][j]);  
 }  
 printf(**"\n"**);  
 }  
  
 **return** 0;  
}

# 16.

*/\*Read a square binary matrix from standard input. First the dimension N is entered (1 < N <= 100) and  
then the elements of the matrix (either 0 or 1) by rows.  
Write a program that will calculate the smallest number of necessary changes of the elements in order  
to transform the matrix to zero diagonal symmetrical matrix. Any transition of an element form 0 to 1  
or form 1 to 0 is counts as a single change. After the conversion all the elements of the main diagonal  
should be 0, and the matrix should be symmetrical with respect to the main diagonal.  
Print the minimal number of transitions required.  
\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>**#include **<math.h>  
  
int** main() {  
 **int** n;  
 scanf(**"%d"**,&n);  
 **int** matrix[100][100];  
 **for**(**int** i=0 ; i<n ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 scanf(**"%d"**,&matrix[i][j]);  
 }  
 }  
 **int** count=0;  
 **for**(**int** i=0 ; i<n ; i++){  
 **if**(matrix[i][i]!=0){  
 count++;  
 }  
 }  
 **for**(**int** i=0 ; i<n ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 **if**(i!=j && matrix[i][j]!=matrix[j][i] && i+j>i+i){  
 count++;  
 }  
 }  
 }  
 printf(**"%d"**,count);  
 **return** 0;  
}

# 17.

#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>**#include **<math.h>  
  
void** writeToFile();  
  
**void** cleanStr(**char** \*str){  
 **int** k=0;  
 **for**(**int** i=0 ; i< strlen(str) ; i++){  
 **if**(isalpha(str[i]) || isspace(str[i])){  
 **if**(k>0){  
 **if**(isspace(str[i]) && isspace(str[k-1])){  
 **continue**;  
 }  
 }  
 str[k]=str[i];  
 k++;  
 }  
 }  
 str[k]=**'\0'**;  
}  
  
**void** printWords(**char** \*str){  
 **for** (**int** i = 0 ; i < strlen(str) ; i++) {  
 **if**(isalpha(str[i])){  
 printf(**"%c"**,str[i]);  
 } **else if**(isspace(str[i])){  
 printf(**"\n"**);  
 }  
 }  
}  
  
**int** main() {  
 writeToFile();  
 FILE \*f= fopen(**"text.txt"**,**"r"**);  
 **char** row[100];  
 **while** (fgets(row,**sizeof**(row),f)!=**NULL**){  
 cleanStr(row);  
 printWords(row);  
 }  
 fclose(f);  
 **return** 0;  
}  
  
**void** writeToFile(){  
 FILE \*f = fopen(**"text.txt"**, **"w"**);  
 **char** c;  
 **while** ((c = getchar()) != **EOF**) {  
 fputc(c, f);  
 }  
 fclose(f);  
}

# 18.

*/\*You are given a text file input.txt containing a single word in each line of the file.  
No words are longer than 20 characters. Write a program that on the SO will print all the  
words from the file that are unsigned hexadecimal numbers (one word per line in uppercase digits).  
At the end also print the total number of such numbers found in the file.  
The hexadecimal numbers can contain both upper and lowercase letters a-f and A-F, but when printing them convert  
them all to uppercase. The possible leading zeroes are also treated as part of the number and should also be printed.  
Since the numbers are unsigned whole numbers no sign or decimal point is considered as part of the number.  
If for example the content of the file input.txt is:  
The\_sum\_is  
64  
and\_the\_minimum\_is  
def  
ined\_as  
a  
035  
where\_signs\_like  
+12AB  
are\_not\_considered  
as\_well\_as  
decimal\_point\_like  
12.f5a  
also\_not  
this1  
or  
2one  
the program should print:  
64  
DEF  
A  
035  
Total: 4  
\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>**#include **<math.h>  
  
void** clearStr(**char** \*str){  
 **if**(str[strlen(str)-1]==**'\n'**){  
 str[strlen(str)-1]=**'\0'**;  
 }  
}  
  
**int** hexNumber(**char** \*word){  
 **for**(**int** i=0 ; i< strlen(word) ; i++){  
 word[i]= toupper(word[i]);  
 }  
 **for**(**int** i=0 ; i< strlen(word) ; i++){  
 **if**(!isalnum(word[i]) && word[i]!=**'\n'**){  
 **return** 0;  
 }  
 **if**(isalpha(word[i])){  
 **if**(word[i]>**'F'** || word[i]<**'A'**){  
 **return** 0;  
 }  
 }  
 }  
 **return** 1;  
}  
  
**int** main() {  
 FILE \*f= fopen(**"C:\\Users\\user\\Desktop\\CCC\\Exams\\input.txt"**,**"r"**);  
 **char** word[100];  
 **int** count=0;  
 **while**(fgets(word,100,f)!=**NULL**){  
 **if**(hexNumber(word) && word[0]!=**'\n'**){  
 printf(**"%s"**,word);  
 count++;  
 }  
 }  
 fclose(f);  
  
 printf(**"Total: %d"**,count);  
 **return** 0;  
}

# 19.

*/\*Write a program that will repeat k times each single occurrence of a vowel in the input file  
"sp.txt" to a new file "output.txt". The first line of the input file contains only the parameter k.  
The first line (containing the parameter k) should not to be written in the output file.\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>**#define **MAX** 100  
  
  
*//ne menuvaj!***void** wtf() {  
 FILE \*f = fopen(**"sp.txt"**, **"w"**);  
 **char** c;  
 **while**((c = getchar()) != **EOF**) {  
 fputc(c, f);  
 }  
 fclose(f);  
}  
**void** rff() {  
 FILE \*f = fopen(**"output.txt"**, **"r"**);  
 **char** c;  
 **while**((c = fgetc(f)) != **EOF**) {  
 putchar(c);  
 }  
 fclose(f);  
}  
**int** isVowel(**char** c){  
 **return** tolower(c)==**'a'** || tolower(c)==**'e'** || tolower(c)==**'i'** || tolower(c)==**'o'** || tolower(c)==**'u'**;  
}  
  
  
**int** main() {  
 wtf();  
 FILE \*fInput= fopen(**"sp.txt"**,**"r"**);  
 FILE \*fOutput= fopen(**"output.txt"**,**"w"**);  
 **int** k;  
 fscanf(fInput,**"%d"**,&k);  
 **char** c;  
 **while**((c= fgetc(fInput))!=**EOF**){  
 **if**(c!=**'\n'**){  
 **if**(isVowel(c)){  
 **for**(**int** i=0 ; i<k-1 ; i++){  
 fprintf(fOutput,**"%c"**,c);  
 }  
 }  
 fprintf(fOutput,**"%c"**,c);  
 }  
 }  
 fclose(fInput);  
 fclose(fOutput);  
  
 rff();  
 **return** 0;  
  
}

# 20.

*/\*In given file datnum.txt find and print the row, where the ratio of digits/letters is largest.  
If there are more than one such rows print the last. No row is longer than 100 characters, and if  
in a row there are no letters, that row is ignored.\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>**#include **<math.h>  
  
int** numDig(**char** \*str){  
 **int** countD=0, countL=0;  
 **for**(**int** i=0 ; i< strlen(str) ; i++){  
 **if**(isdigit(str[i])){  
 countD++;  
 }  
 **if**(isalpha(str[i])){  
 countL++;  
 }  
 }  
 **return** countD!=0 && countL!=0;  
}  
  
**float** ratio(**char** \*str){  
 **int** countL=0, countD=0;  
 **for**(**int** i=0 ; i< strlen(str) ; i++){  
 **if**(isdigit(str[i])){  
 countD++;  
 }  
 **if**(isalpha(str[i])){  
 countL++;  
 }  
 }  
 **return** (**float**)countD/(**float**)countL;  
}  
  
**void** clearString (**char** \* string) {  
 **if** (string[strlen(string)-1]==**'\n'**){  
 string[strlen(string)-1]=**'\0'**;  
 }  
}  
  
**int** main() {  
 FILE \*f= fopen(**"C:\\Users\\user\\Desktop\\CCC\\Exams\\datnum.txt"**,**"r"**);  
 **char** row[100];  
 **int** flag=1;  
 **float** max,Ratio;  
 **int** maxL=0;  
 **while**(fgets(row,100,f)!=**NULL**){  
 clearString(row);  
 **if**(numDig(row)==0){  
 **continue**;  
 }  
 Ratio= ratio(row);  
 **if**(flag){  
 max=Ratio;  
 maxL= strlen(row);  
 flag=0;  
 } **else if**(Ratio>=max){  
 max=Ratio;  
 maxL= strlen(row);  
 }  
 }  
 FILE \*fp= fopen(**"C:\\Users\\user\\Desktop\\CCC\\Exams\\datnum.txt"**,**"r"**);  
 **char** line[100];  
 **while**(fgets(line,100,fp)!=**NULL**){  
 clearString(line);  
 Ratio= ratio(line);  
 **if**(Ratio==max && maxL== strlen(line)){  
 puts(line);  
 }  
 }  
 fclose(f);  
 fclose(fp);  
 **return** 0;  
}

# 21.

*/\*From SI are read dimension and elements of square matrix (MAX 100x100) of integers.  
Create a new matrix where each element of the main diagonal will be replaced with the sum  
of all elements from the same row right from the element and elements from the same column over the element.  
Also replace each element of the antidiagonal with the sum of all elements from the same row before the element  
and elements from the same column below the element.  
The blue red elements are part of the main diagonal, the blue elements are part of antidiagonal,  
and the element in purple is part of both diagonals.  
If there is no such elements, the element is 0. If some element is on both diagonals, both sums are summed.\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>**#include **<math.h>  
  
  
int** main() {  
 **int** n;  
 **int** matrix[100][100];  
 scanf(**"%d"**,&n);  
 **for**(**int** i=0 ; i<n ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 scanf(**"%d"**,&matrix[i][j]);  
 }  
 }  
 **int** newMatrix[100][100];  
 **for**(**int** i=0 ; i<n ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 newMatrix[i][j]=matrix[i][j];  
 }  
 }  
 **for**(**int** i=0 ; i<n ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 **if**(i==j){  
 **int** sumRowRight=0;  
 **int** sumColumnOver=0;  
 **for**(**int** k=j+1 ; k<n ; k++){  
 sumRowRight+=matrix[i][k];  
 }  
 **for**(**int** k=i-1 ; k>=0 ; k--){  
 sumColumnOver+=matrix[k][j];  
 }  
 newMatrix[i][j]=sumColumnOver+sumRowRight;  
 }  
 **if**(i+j==n-1){  
 **int** sumRowLeft=0;  
 **int** sumColumnBelow=0;  
 **for**(**int** k=j-1 ; k>=0 ; k--){  
 sumRowLeft+=matrix[i][k];  
 }  
 **for**(**int** k=i+1 ; k<n ; k++){  
 sumColumnBelow+=matrix[k][j];  
 }  
 newMatrix[i][j]=sumColumnBelow+sumRowLeft;  
 }  
 **if**(i==j && i+j==n-1){  
 **int** sumRowRight=0;  
 **int** sumColumnOver=0;  
 **int** sumRowLeft=0;  
 **int** sumColumnBelow=0;  
 **for**(**int** k=j+1 ; k<n ; k++){  
 sumRowRight+=matrix[i][k];  
 }  
 **for**(**int** k=i-1 ; k>=0 ; k--){  
 sumColumnOver+=matrix[k][j];  
 }  
 **for**(**int** k=j-1 ; k>=0 ; k--){  
 sumRowLeft+=matrix[i][k];  
 }  
 **for**(**int** k=i+1 ; k<n ; k++){  
 sumColumnBelow+=matrix[k][j];  
 }  
 newMatrix[i][j]=sumColumnBelow+sumRowLeft+sumRowRight+sumColumnOver;  
 }  
 }  
  
 }  
 **for**(**int** i=0 ; i<n ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 printf(**"%d "**,newMatrix[i][j]);  
 }  
 printf(**"\n"**);  
 }  
 **return** 0;  
}

# 22.

*/\*Write a program that will read from SI array of natural numbers (not more than 100). Print all the elements of  
the array that represent strong numbers (each in new line). One number is strong if the sum of factorials of each  
of his digits is equal to the number itself.  
Example, number 145 is strong because:  
145 = 1! + 4! + 5!  
If a number is strong should be checked in a separate function, and for computing factorial a separate recursive  
function should be implemented and used.\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>**#include **<math.h>  
  
int** factorial(**int** n){  
 **if**(n==0){  
 **return** 1;  
 } **else**{  
 **return** n\* factorial(n-1);  
 }  
}  
  
**int** strong(**int** n){  
 **int** sum=0;  
 **int** tmp=n;  
 **while**(tmp){  
 **int** ld=tmp%10;  
 sum+= factorial(ld);  
 tmp/=10;  
 }  
 **return** sum==n;  
}  
  
**int** main() {  
 **int** n;  
 scanf(**"%d"**,&n);  
 **int** array[100];  
 **for**(**int** i=0 ; i<n ; i++){  
 scanf(**"%d"**,&array[i]);  
 }  
 printf(**"Strong numbers:\n"**);  
 **for**(**int** i=0 ; i<n ; i++){  
 **if**(strong(array[i])){  
 printf(**"%d\n"**,array[i]);  
 }  
 }  
 **return** 0;  
}

# 23.

*/\*Read a an array of N integers from standard input (first read the number N than the elements).  
Than call a recursive function on the array that will change on the following way:  
where max is a function that returns the maximum of the two numbers.  
The function should return the number of switched elements. Then print that number and the changed array.  
\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>**#include **<math.h>  
  
int** max(**int** a, **int** b){  
 **int** max;  
 **if**(a>b){  
 max=a;  
 } **else**{  
 max=b;  
 }  
 **return** max;  
}  
  
**void** swap(**int** array[], **int** n){  
 **for**(**int** i=0 ; i<n/2 ; i++){  
 array[i]= max(array[i],array[n-i-1]);  
 array[n-i-1]=max(array[i],array[n-i-1]);  
 }  
}  
  
**void** swapR(**int** array[], **int** n, **int** m){  
 **if**(n==0){  
 array[0]=max(array[0],array[m]);  
 } **else**{  
 array[n]=max(array[m-n],array[n]);  
 swapR(array,n-1,m);  
 }  
}  
  
**int** main() {  
 **int** n;  
 scanf(**"%d"**,&n);  
 **int** array[100];  
 **for**(**int** i=0 ; i<n ; i++){  
 scanf(**"%d"**,&array[i]);  
 }  
 **int** tmp[100];  
 **for**(**int** i=0 ; i<n ; i++){  
 tmp[i]=array[i];  
 }  
 swapR(array,n-1,n-1);  
 **int** count=0;  
 **for**(**int** i=0 ; i<n/2 ; i++){  
 **if**(array[i]!=tmp[i]){  
 count++;  
 }  
 }  
 printf(**"%d\n"**,count);  
 **for**(**int** i=0 ; i<n ; i++){  
 printf(**"%d "**,array[i]);  
 }  
 **return** 0;  
}

# 24.

*/\*Write a fully recursive function triangle (n) which given the input argument n,  
prints a triangle from the numbers from 1 to n. Additional functions can be used, but no loops should be used.  
Example for n=4.  
 1  
 12  
 123  
 1234  
\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>**#include **<math.h>  
  
void** triangle (**int** n,**int** m){  
 **if**(n==0){  
 **return**;  
 } **else**{  
 printf(**"%d"**,m-n+1);  
 **return** triangle(n-1,m);  
 }  
}  
  
**void** triangle2 (**int** n,**int** m){  
 **if**(n==0){  
 **return**;  
 } **else**{  
 triangle(m-n+1,m-n+1);  
 printf(**"\n"**);  
 **return** triangle2(n-1,m);  
 }  
}  
  
**int** main(){  
 **int** n;  
 scanf(**"%d"**,&n);  
 triangle2 (n,n);  
 **return** 0;  
}

# 25.

*/\*N positive integers are entered from the standard input and stored in an array a (N <100)  
\*\*. Also, a single-digit integer \*\*k\*\* is entered. Write a function that will find the maximum of  
the abbreviated numbers of the elements in the array \*\*a\*\* and provide a call to that function in  
the main () function. A given element of the array is abbreviated as follows:  
all digits smaller than or equal to \*\*k are deleted from the number.  
  
For example: If k = 3 and a[0] = 12345, then abbreviated a[0] becomes 45, if a[1] = 458763,  
a[1] becomes 45876, and a[2] = 112 is shortened to a[2] = 0. The maximum abbreviated number is 45876.\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>  
  
int** countOfDigits(**int** number){  
 **int** counter=1;  
 **while**(number){  
 counter\*=10;  
 number/=10;  
 }  
 counter/=10;  
 **return** counter;  
}  
  
**int** abbreviated (**int** n, **int** k){  
 **int** tmp=n;  
 **int** abbrev=0;  
 **while**(tmp){  
 **int** d= countOfDigits(tmp);  
 **int** fd=tmp/d;  
 **if**(fd>k){  
 abbrev= abbrev\*10+fd;  
 }  
 tmp%=d;  
 }  
 **return** abbrev;  
}  
**int** maxAbbElement(**int** a[], **int** n, **int** k){  
 **int** max= abbreviated(a[0],k);  
 **for**(**int** i=1 ; i<n ; i++){  
 **int** abb= abbreviated(a[i],k);  
 **if**(abb>max){  
 max= abb;  
 }  
 }  
 **return** max;  
}  
  
**int** main() {  
 **int** n;  
 scanf(**"%d"**,&n);  
 **int** a[100];  
 **for**(**int** i=0 ; i<n ; i++){  
 scanf(**"%d"**,&a[i]);  
 }  
 **int** k;  
 scanf(**"%d"**,&k);  
 **int** max=maxAbbElement(a, n, k);  
 printf(**"%d"**,max);  
 **return** 0;  
}

# 26.

*/\*N positive integers are entered from the standard input and are stored in an array (N <100).  
All elements that meet the next criteria should be presented on the standard output in the same  
order that they have appeared on the standard input. The criteria is following: the most important  
digit (the digit on the first place) is odd and the least important digit is even. If, no such number  
was entered, you should print out "No such elements!" message.  
  
The implementation of checking the criteria should be performed in a separate  
function (recursive function for additional 5 points).\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>**#include **<math.h>  
  
int** countOfDig(**int** n){  
 **int** count=1;  
 **while**(n){  
 count\*=10;  
 n/=10;  
 }  
 count/=10;  
 **return** count;  
}  
**int** criteria(**int** n){  
 **int** d= countOfDig(n);  
 **int** fd=n/d;  
 **int** ld=n%10;  
 **return** fd%2!=0 && ld%2==0;  
}  
  
**int** main() {  
 **int** n;  
 scanf(**"%d"**,&n);  
 **int** array[100];  
 **for**(**int** i=0 ; i<n ; i++){  
 scanf(**"%d"**,&array[i]);  
 }  
 **int** flag=1;  
 **for**(**int** i=0 ; i<n ; i++){  
 **if**(criteria(array[i])){  
 printf(**"%d\n"**,array[i]);  
 flag=0;  
 }  
 }  
 **if**(flag){  
 printf(**"No such elements!"**);  
 }  
 **return** 0;  
}

# 27.

*/\*Read the dimension N (with a maximum value of 100) and the values (possible values are only 1 and 0)  
of one square matrix $A\_ {NxN}$ from the standard input. Create a new matrix $B\_ {NxN}$ in the following way:  
For each element in B, write the value of the same corresponding element from A. During this procedure,  
if in the neighborhood of one element of A (neighbors are the elements above it or below it, on the left  
or right side of it) there are minimum 3 elements with a value of 1, then that element in the matrix B gets  
a value of 1 (although in A the corresponding value was zero).  
Print the newly obtained matrix B.  
\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>**#include **<math.h>  
  
  
int** main(){  
 **int** n;  
 scanf(**"%d"**,&n);  
 **int** A[100][100],B[100][100];  
  
 **for**(**int** i=0 ; i<n ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 scanf(**"%d"**,&A[i][j]);  
 }  
 }  
 **for**(**int** i=0 ; i<n ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 **if**(A[i][j]==1){  
 B[i][j]=1;  
 } **else**{  
 **if**((A[i][j-1]==1 && A[i-1][j]==1 && A[i][j+1]==1) || (A[i-1][j]==1 && A[i][j+1]==1 && A[i+1][j]==1) || (A[i][j-1]==1 && A[i-1][j]==1 && A[i+1][j]) || (A[i][j-1]==1 && A[i][j+1]==1 && A[i+1][j])){  
 B[i][j]=1;  
 } **else**{  
 B[i][j]=0;  
 }  
 }  
 }  
 }  
 **for**(**int** i=0 ; i<n ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 printf(**"%d "**,B[i][j]);  
 }  
 printf(**"\n"**);  
 }  
 **return** 0;  
}

# 28.

*/\*Firstly read the dimension of one matrix (m, n <= 100) from the standard input, and then the elements of the matrix.  
Generate an array (with at most m x n elements) of the elements of the matrix that will contain the elements of each  
row that appear exactly once in that row. The order of writing the elements in the array corresponds to the order of  
the rows and the order of the elements within the row. If the newly generated array has no elements, print N, otherwise,  
print the elements of the array.\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>**#include **<math.h>  
  
  
int** main(){  
 **int** m,n;  
 scanf(**"%d%d"**,&m,&n);  
 **int** matrix[100][100],array[100];  
  
 **for**(**int** i=0 ; i<m ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 scanf(**"%d"**,&matrix[i][j]);  
 }  
 }  
 **int** t=0;  
 **int** flag2=1;  
 **for**(**int** i=0 ; i<m ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 **int** flag=1;  
 **for**(**int** k=0 ; k<n ; k++){  
 **if**(k!=j && matrix[i][j]==matrix[i][k]){  
 flag=0;  
 **break**;  
 }  
 }  
 **if**(flag){  
 array[t]=matrix[i][j];  
 t++;  
 flag2=0;  
 }  
 }  
 }  
 **if**(flag2){  
 printf(**"N"**);  
 } **else**{  
 **for**(**int** i=0 ; i<t ; i++){  
 printf(**"%d "**,array[i]);  
 }  
 }  
 **return** 0;  
}

# 29.

*/\*Read the dimensions M and N (not bigger than 100) and the values (whole numbers)  
of one matrix A\_ {MxN} from the standard input. Print out the index of the first  
column that has the smallest sum of all of its elements.\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>**#include **<math.h>  
  
  
int** main(){  
 **int** m,n;  
 scanf(**"%d%d"**,&m,&n);  
 **int** matrix[100][100];  
  
 **for**(**int** i=0 ; i<m ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 scanf(**"%d"**,&matrix[i][j]);  
 }  
 }  
 **int** sum,minSum,minIndex;  
 **int** flag=1;  
 **for**(**int** j=0 ; j<n ; j++){  
 sum=0;  
 **for**(**int** i=0 ; i<m ; i++){  
 sum+=matrix[i][j];  
 }  
 **if**(flag){  
 minSum=sum;  
 minIndex=j;  
 flag=0;  
 } **else if**(sum<minSum){  
 minSum=sum;  
 minIndex=j;  
 }  
 }  
 printf(**"%d"**,minIndex);  
 **return** 0;  
}

# 30.

*/\*A text file "input.txt" consist of digits and letters. The number of digits and letters  
in a row is limited to 80. Write a program that for every row will print out the letters  
concatenated next to each other first followed by the sum of all numbers that appear in that row.  
If there are no digits in the row, for the sum of the numbers you should print out 0.\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>**#include **<math.h>***//ne menuvaj!***void** wtf() {  
 FILE \*f = fopen(**"input.txt"**, **"w"**);  
 **char** c;  
 **while**((c = getchar()) != **'#'**) {  
 fputc(c, f);  
 }  
 fclose(f);  
}  
  
**int** isNumber(**char** \*str){  
 **int** sum=0;  
 **int** currentNumber=0;  
 **for**(**int** i=0 ; i< strlen(str) ; i++){  
 **if**(isdigit(str[i])){  
 currentNumber=10\*currentNumber+(str[i]-**'0'**);  
 } **else**{  
 sum+=currentNumber;  
 currentNumber=0;  
 }  
 }  
 **return** sum;  
}  
**void** letters(**char** \*str){  
 **for**(**int** i=0 ; i< strlen(str) ; i++){  
 **if**(isalpha(str[i])){  
 printf(**"%c"**,str[i]);  
 }  
 }  
}  
  
**int** main() {  
 wtf();  
 FILE \*f= fopen(**"input.txt"**,**"r"**);  
 **char** row[100];  
 **while** (fgets(row,**sizeof**(row),f)!=**NULL**){  
 letters(row);  
 **int** sum= isNumber(row);  
 printf(**"%d\n"**,sum);  
 }  
 fclose(f);  
 **return** 0;  
}

# 31.

#include **<stdio.h>**#include **<ctype.h>**#include **<string.h>**#define **MAX** 31  
  
**void** writeToFile(**char** \* file) {  
 FILE \*f = fopen(file, **"w"**);  
 **char** c;  
 **while**((c = getchar()) != **'#'**) {  
 fputc(c, f);  
 }  
 fclose(f);  
}  
  
**int** isVowel(**char** c){  
 **return** tolower(c)==**'a'** || tolower(c)==**'e'** || tolower(c)==**'i'** || tolower(c)==**'o'** || tolower(c)==**'u'**;  
}  
  
**int** isPalindrome(**char** \*word){  
 **for**(**int** i=0 ; i< strlen(word) ; i++){  
 **if**(tolower(word[i])!= tolower(word[strlen(word)-1-i])){  
 **return** 0;  
 }  
 }  
 **return** 1;  
}  
  
**int** vowel(**char** \*word){  
 **int** k=0;  
 **char** vowel[100];  
 **for**(**int** i=0 ; i< strlen(word) ; i++){  
 **if**(isVowel(word[i])){  
 vowel[k]=word[i];  
 k++;  
 }  
 }  
 vowel[k]=**'\0'**;  
 **return** isPalindrome(vowel);  
}  
  
**void** clearStr(**char** \*str){  
 **if**(str[strlen(str)-1]==**'\n'**){  
 str[strlen(str)-1]=**'\0'**;  
 }  
}  
  
**int** main() {  
 writeToFile(**"input.txt"**);  
 FILE \*f= fopen(**"input.txt"**,**"r"**);  
 **char** word[100];  
 **while** (fgets(word,100,f)!=**NULL**){  
 clearStr(word);  
 **if**(vowel(word)){  
 puts(word);  
 }  
 }  
 fclose(f);  
 **return** 0;  
}

# 32.

*/\*A text file "rectangle.in" is given. In the first row of the file there is an integer N, and in the second row,  
an array of N single-digit positive numbers. Find the numbers A and B that repeat the most times in the array.  
Then, print the rectangle with dimension A x B on the standard output, in the following format:  
\*\*.....\*  
\*\*.....\*  
\*\*.....\*  
\*\*.....\*  
\*\*.....\*  
(A lines with B-stars, where A is the smaller of the two numbers found).  
In the test cases there are always exactly two numbers that appear the same most times.\*/*#include **<stdio.h>**#include **<string.h>***//ne menuvaj!* **void** wtf() {  
 FILE \*f = fopen(**"rectangle.in"**, **"w"**);  
 **char** c;  
 **while**((c = getchar()) != **EOF**) {  
 fputc(c, f);  
 }  
 fclose(f);  
 }  
  
 **int** main() {  
 wtf();  
 *// vashiot kod ovde* FILE \*f= fopen(**"rectangle.in"**,**"r"**);  
 **int** n;  
 fscanf(f,**"%d"**,&n);  
 **int** array[100];  
 **for**(**int** i=0 ; i<n ; i++){  
 fscanf(f,**"%d"**,&array[i]);  
 }  
 **int** maxE1,maxE2;  
 **int** count0=0,count1=0;  
 **for**(**int** i=0 ; i<n ; i++){  
 **if**(array[0]==array[i]){  
 count0++;  
 }  
 **if**(array[1]==array[i]){  
 count1++;  
 }  
 }  
 **int** maxC1,maxC2;  
 **if**(array[0]==array[1]){  
 maxE1=array[0];  
 maxE2=0;  
 count1=0;  
 maxC1=count0;  
 maxC2=0;  
 }  
  
 **if**(count0==count1 && array[0]>array[1]){  
 maxE2=array[1];  
 maxE1=array[0];  
 maxC2=count1;  
 maxC1=count0;  
 } **else if**(count0==count1 && array[0]<array[1]){  
 maxE2=array[0];  
 maxE1=array[1];  
 maxC2=count0;  
 maxC1=count1;  
 }  
 **if**(count0>count1){  
 maxE1=array[0];  
 maxE2=array[1];  
 maxC1=count0;  
 maxC2=count1;  
 } **else if**(count1>count0){  
 maxE1=array[1];  
 maxE2=array[0];  
 maxC1=count1;  
 maxC2=count0;  
 }  
 **for**(**int** i=2 ; i<n ; i++){  
 **if**(array[i]==maxE1 || array[i]==maxE2){  
 **continue**;  
 }  
 **int** counter=0;  
 **for**(**int** j=0 ; j<n ; j++){  
 **if**(array[i]==array[j]){  
 counter++;  
 }  
 }  
 **if**(counter==maxC1 && array[i]>maxE1){  
 maxE2=maxE1;  
 maxE1=array[i];  
 maxC2=maxC1;  
 maxC1=counter;  
 }  
  
 **if**(counter>maxC1){  
 maxE2=maxE1;  
 maxE1=array[i];  
 maxC2=maxC1;  
 maxC1=counter;  
 } **else if**(counter>maxC2){  
 maxE2=array[i];  
 maxC2=counter;  
 }  
 **if**(maxE1==maxE2){  
 maxE2=0;  
 maxC2=0;  
 }  
  
 }  
 **for**(**int** i=0 ; i<maxE2; i++){  
 **for**(**int** j=0 ; j<maxE1 ; j++){  
 printf(**"\* "**);  
 }  
 printf(**"\n"**);  
 }  
  
 }

# 33.

*/\*Write a program that will print on SO the contents of given textual file "text.txt".  
While printing, in front of each row should print the ratio of upper/lower case letters in the row.  
Maximal length of the row is 80 characters. At the end print the number of the row with largest ratio of upper/lower case letters.\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>  
  
void** wtf() {  
 FILE \*f = fopen(**"text.txt"**, **"w"**);  
 **char** c;  
 **while**((c = getchar()) != **EOF**) {  
 fputc(c, f);  
 }  
 fclose(f);  
}  
  
**float** ratioUpperLower(**char** \*str){  
 **float** countUpper=0, countLower=0;  
 **for**(**int** i=0 ; i< strlen(str) ; i++){  
 **if**(isalpha(str[i])){  
 **if**(isupper(str[i])){  
 countUpper++;  
 } **else**{  
 countLower++;  
 }  
 }  
 }  
 **return** countUpper/countLower;  
}  
**void** clearStr(**char** \*str){  
 **if**(str[strlen(str)-1]==**'\n'**){  
 str[strlen(str)-1]=**'\0'**;  
 }  
}  
  
**int** main() {  
 wtf();  
 FILE \*f= fopen(**"text.txt"**,**"r"**);  
 **char** row[100];  
 **int** flag=1,maxRow,k=0;  
 **float** maxRatio;  
 **while**(fgets(row,**sizeof**(row),f)!=**NULL**){  
 clearStr(row);  
 **float** ratio= ratioUpperLower(row);  
 printf(**"%.2f "**,ratio);  
 puts(row);  
 **if**(flag){  
 maxRow=k;  
 maxRatio=ratio;  
 flag=0;  
 } **else if**(ratio>maxRatio){  
 maxRow=k;  
 maxRatio=ratio;  
 }  
 k++;  
 }  
 printf(**"%d"**,maxRow);  
 fclose(f);  
 **return** 0;  
}

# 34.

*/\*The file \* input.txt \* is given and contains a single word that is not longer than 20 letters in each line.  
Write a program that will print the word that has the biggest number of different letters. Words that have less than  
four letters are not taken into account during the check. Do not make a difference between upper and lower case letters.  
If there are more words that meet the criteria, the last one is printed.\*/*#include **<stdio.h>**#include **<ctype.h>**#include **<string.h>  
  
void** writeToFile(**char** \* file) {  
 FILE \*f = fopen(file, **"w"**);  
 **char** c;  
 **while**((c = getchar()) != **'#'**) {  
 fputc(c, f);  
 }  
 fclose(f);  
}  
**int** diffLetter(**char** \*str){  
 **int** counter=0;  
 **for**(**int** i=0 ; i< strlen(str) ; i++){  
 **for**(**int** j=i+1 ; j< strlen(str) ; j++){  
 **if**(tolower(str[i])!= tolower(str[j])){  
 counter++;  
 }  
 }  
 }  
 **return** counter;  
}  
  
**void** clearStr(**char** \*str){  
 **for**(**int** i=0 ; i< strlen(str) ; i++){  
 **if**(str[strlen(str)-1]==**'\n'**){  
 str[strlen(str)-1]=**'\0'**;  
 }  
 }  
}  
  
**int** main() {  
 writeToFile(**"input.txt"**);  
 FILE \*f= fopen(**"input.txt"**,**"r"**);  
 **char** word[21];  
 **int** flag=1;  
 **int** maxC;  
 **char** maxWord[21];  
 **while**(fgets(word,21,f)!=**NULL**){  
 **if**(strlen(word)<4){  
 **continue**;  
 }  
 clearStr(word);  
 **int** counter= diffLetter(word);  
 **if**(flag){  
 maxC=counter;  
 **for**(**int** i=0 ; i< strlen(word) ; i++){  
 maxWord[i]=word[i];  
 }  
 flag=0;  
 } **else if**(counter>=maxC){  
 maxC=counter;  
 **for**(**int** i=0 ; i< strlen(word) ; i++){  
 maxWord[i]=word[i];  
 }  
 }  
 }  
 puts(maxWord);  
 fclose(f);  
 **return** 0;  
}

# 35.

*/\*Z-diagonal of given square matrix is composed of the elements of the first row, the elements of the supportive diagonal,  
and the elements of the last row (marked with blue color on the example).  
From SI read the dimensions of square matrix (integer N, N > 2, N<100),  
and the elements of the matrix (real numbers). Store the elements of the Z-diagonal in a array,  
starting from the first row, continuing with the supportive diagonal, and finishing with the last row  
(the last element of the first row is the same element with the first element of the supportive diagonal,  
and the last element of the supportive diagonal is the same as the first element of the last row, and  
these elements should be stored only once). The resulted array should be printed on the SO.  
Then print the original matrix, but with the elements of Z-diagonal in reverse order.\*/*#include **<stdio.h>**#include **<ctype.h>**#include **<string.h>  
  
int** main(){  
 **int** n;  
 scanf(**"%d"**,&n);  
 **double** matix[100][100];  
 **for**(**int** i=0 ; i<n ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 scanf(**"%lf"**,&matix[i][j]);  
 }  
 }  
 **int** k=0;  
 **double** array[100];  
 **for**(**int** i=0 ; i<n ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 **if**(i==0 && j<n-1){  
 array[k]=matix[i][j];  
 k++;  
 }  
 **if**(i+j==n-1){  
 array[k]=matix[i][j];  
 k++;  
 }  
 **if**(i==n-1 && j>0){  
 array[k]=matix[i][j];  
 k++;  
 }  
 }  
 }  
  
 **int** h=k-1;  
 **for**(**int** i=0 ; i<n ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 **if**(i==0 && j<n-1){  
 matix[i][j]=array[h];  
 h--;  
 }  
 **if**(i+j==n-1){  
 matix[i][j]=array[h];  
 h--;  
 }  
 **if**(i==n-1 && j>0){  
 matix[i][j]=array[h];  
 h--;  
 }  
 }  
 }  
 **for**(**int** i=0 ; i<k ; i++){  
 printf(**"%.2lf "**,array[i]);  
 }  
 printf(**"\n"**);  
 **for**(**int** i=0 ; i<n ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 printf(**"%.2lf "**,matix[i][j]);  
 }  
 printf(**"\n"**);  
 }  
 **return** 0;  
}

# 36.

*/\*A whole number x ( 1<=x <=10) is read from SI.  
After that, an array of no more than 100 integers is read. First the count n of numbers to follow is given, and then the n numbers.  
Write a program that will print in descending order the numbers from the array that contain exactly x unique digits.  
The calculation of the count of unique digits for a given number should be performed in a separate function.  
Solutions without functions will be graded with a maximum of 40% of the points.  
\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>  
  
int** uniqueDigits(**int** number, **int** x){  
 **int** counterArray[10]={0,0,0,0,0,0,0,0,0,0};  
 **while**(number){  
 **int** ld=number%10;  
 counterArray[ld]++;  
 number/=10;  
 }  
 **int** result=0;  
 **for**(**int** i=0 ; i<10 ; i++){  
 **if**(counterArray[i]>0){  
 result++;  
 }  
 }  
 **return** result==x;  
}  
  
**void** bubbleSort(**int** a[],**int** n){  
 **for**(**int** i=0 ; i<n ; i++){  
 **for**(**int** j=0 ; j<n-1-i ; j++){  
 **if**(a[j]<a[j+1]){  
 **int** tmp = a[j];  
 a[j] = a[j + 1];  
 a[j + 1] = tmp;  
 }  
 }  
 }  
}  
  
**int** main() {  
 **int** x;  
 scanf(**"%d"**,&x);  
 **int** n;  
 scanf(**"%d"**,&n);  
  
 **int** array[100];  
 **int** k=0;  
 **for**(**int** i=0 ; i<n ; i++){  
 **int** number;  
 scanf(**"%d"**,&number);  
 **if**(uniqueDigits(number,x)){  
 array[k]=number;  
 k++;  
 }  
 }  
 bubbleSort(array,k);  
 **for**(**int** i=0 ; i<k ; i++){  
 printf(**"%d "**,array[i]);  
 }  
 **return** 0;  
}

# 37.

*/\*A text file named “input.txt” is given. The lines in the file contain no more than 100 characters.  
Write a program that for each line of the file will print information for the number of letters,  
number of words, and the line itself. Words are considered all combinations of letters and digits  
separated with space or punctuation marks.  
The printing format is the following:  
b, w: row  
Where b is the count of letters and digits, w is the count of words and row is the content of the line.  
  
\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>  
  
void** clearStr(**char** \*str){  
 **if**(str[strlen(str)-1]==**'\n'**){  
 str[strlen(str)-1]=**'\0'**;  
 }  
}  
  
**void** proccesRow(**char** \*str){  
 **int** countLD=0;  
 **int** spaces=0;  
 **for**(**int** i=0 ; i< strlen(str) ; i++){  
 **if**(isalnum(str[i])){  
 countLD++;  
 }  
 **if**(isspace(str[i])){  
 spaces++;  
 }  
 }  
 printf(**"%d, %d: %s\n"**,countLD,spaces+1,str);  
}  
  
**int** main() {  
 FILE \*f= fopen(**"C:\\Users\\user\\Desktop\\CCC\\Exams\\input.txt"**,**"r"**);  
 **char** row[100];  
 **while**(fgets(row,**sizeof**(row),f)!=**NULL**){  
 clearStr(row);  
 proccesRow(row);  
 }  
 fclose(f);  
}

# 38.

*/\*The file matrica.txt contains the elements of an integer matrix (A).  
The number of rows n and the number of columns m of the matrix (n, m < 120) are written in the first line of the file.  
After them, in each line of the file, the elements of each row of the matrix are written.  
Print the number of index numbers in each column.  
An element in a matrix is considered for index number if its value is identical with the value that  
is obtained by concatenation of the index of the row and the index of the column of the element.  
Help:  
Index element: The value of the element Aij is the same with ij as a number.  
• If i=10, j=31, the element Aij should have the value of 1031 in order to be an index number.  
\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>  
  
int** concatenated(**int** i, **int** j){  
 **if**(j<10){  
 **return** i\*10+j;  
 }  
 **if**(j<100){  
 **return** i\*100+j;  
 }  
 **if**(j<1000){  
 **return** i\*1000+j;  
 }  
}  
  
**int** indexElement(**int** i, **int** j, **int** element){  
 **return** concatenated(i,j)==element;  
}  
  
**int** main() {  
 FILE \*f= fopen(**"C:\\Users\\user\\Desktop\\CCC\\Exams\\matrica.txt"**,**"r"**);  
 **int** m,n;  
 fscanf(f,**"%d%d"**,&m,&n);  
  
 **int** matrix[120][120];  
 **for**(**int** i=0 ; i<m ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 fscanf(f,**"%d"**,&matrix[i][j]);  
 }  
 }  
 **for**(**int** j=0 ; j<n ; j++){  
 **int** counter=0;  
 **for**(**int** i=0 ; i<m ; i++){  
 **if**(indexElement(i,j,matrix[i][j])){  
 counter++;  
 }  
 }  
 printf(**"%d\n"**,counter);  
 }  
 fclose(f);  
 **return** 0;  
}

# 39.

*/\*A number N (N<100) and then N strings are read from SI. The strings characters contain letters,  
digits and special characters and each of them contains no more than 50 characters.  
Write a program that will print on the screen all the strings in which the substring  
A1c (A1C, a1C, a1c the case doesn’t matter) appears at least 2 times.  
The printing should be done in the same order in which the strings were read.  
Also, when printing the strings, all letters should be printed in lowercase.  
\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>  
  
void** toLower(**char** \*str){  
 **for**(**int** i=0 ; i< strlen(str) ; i++){  
 **if**(isalpha(str[i])){  
 str[i]= tolower(str[i]);  
 }  
 }  
}  
  
**int** isSubString(**char** \*str){  
 **char** \*result= strstr(str,**"a1c"**);  
  
 **return** strstr(result,**"a1c"**)!=**NULL**;  
}  
  
**int** main() {  
 **int** n;  
 scanf(**"%d\n"**,&n);  
 **for**(**int** i=0 ; i<n ; i++){  
 **char** str[50];  
 fgets(str,**sizeof**(str),**stdin**);  
 str[strlen(str)-1]=**'\0'**;  
 toLower(str);  
 **if**(isSubString(str)){  
 printf(**"%s\n"**,str);  
 }  
 }  
 **return** 0;  
}

# 40.

#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>  
  
  
  
int** main() {  
 **int** m,n;  
 scanf(**"%d%d"**,&m,&n);  
 **int** k,r;  
 scanf(**"%d%d"**,&r,&k);  
 **int** A[100][100];  
 **for** (**int** i=0 ; i<m ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 scanf(**"%d"**,&A[i][j]);  
 }  
 }  
 **int** min;  
 min=A[0][0];  
 **for** (**int** i=0 ; i<m ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 **if**(A[i][j]<min){  
 min=A[i][j];  
 }  
 }  
 }  
 **for** (**int** i=0 ; i<m ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 **if**(i<r && j<k){  
 A[i][j]=min;  
 }  
 }  
 }  
 **for** (**int** i=0 ; i<m ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 printf(**"%d "**,A[i][j]);  
 }  
 printf(**"\n"**);  
 }  
 **return** 0;  
}

# 41.

#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>  
  
int** shouldSwap(**int** array[], **int** n, **int** \*indexMax1, **int** \*indexMax2){  
 **int** max1,max2;  
 **if**(array[0]>array[1]){  
 max1=array[0];  
 max2=array[1];  
 \*indexMax1=0;  
 \*indexMax2=1;  
 } **else**{  
 max1=array[1];  
 max2=array[0];  
 \*indexMax1=1;  
 \*indexMax2=0;  
 }  
 **for**(**int** i=2 ; i<n ; i++){  
 **if**(array[i]>max1){  
 max2=max1;  
 max1=array[i];  
 \*indexMax2=\*indexMax1;  
 \*indexMax1=i;  
 } **else if**(array[i]>max2){  
 max2=array[i];  
 \*indexMax2=i;  
 }  
 }  
 **return** \*indexMax1>\*indexMax2;  
}  
  
**int** main() {  
 **int** n;  
 scanf(**"%d\n"**,&n);  
 **int** array[100];  
 **for**(**int** i=0 ; i<n ; i++){  
 scanf(**"%d"**,&array[i]);  
 }  
 **int** indx1,indx2;  
 **if**(shouldSwap(array,n,&indx1,&indx2)){  
 **int** tmp=array[indx1];  
 array[indx1]=array[indx2];  
 array[indx2]=tmp;  
 }  
 **for**(**int** i=0 ; i<n ; i++){  
 printf(**"%d "**,array[i]);  
 }  
 **return** 0;  
}

# 42.

#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>  
  
  
  
int** main() {  
 FILE \*fInput= fopen(**"C:\\Users\\user\\Desktop\\CCC\\Exams\\input.txt"**,**"r"**);  
 **int** n;  
 fscanf(fInput,**"%d"**,&n);  
 **int** matrix[100][100];  
 **for**(**int** i=0 ; i<n ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 fscanf(fInput,**"%d"**,&matrix[i][j]);  
 }  
 }  
 fclose(fInput);  
 **int** tmp[4];  
 **for**(**int** i=0 ; i<n/2 ; i++){  
 **int** j=n-1-i;  
 tmp[0]=matrix[i][i]; *//top left corner* tmp[1]=matrix[i][j]; *//top right cornenr* tmp[2]=matrix[j][i]; *//bottom left corner* tmp[3]=matrix[j][j]; *//bottom right corner* matrix[i][i]=tmp[2]; *//top left = bottom left* matrix[i][j]=tmp[0]; *//top right = top left* matrix[j][j]=tmp[1]; *//botom right = top right* matrix[j][i]=tmp[3]; *//bottom left = bottom right* }  
 FILE \*fOutput= fopen(**"C:\\Users\\user\\Desktop\\CCC\\Exams\\output.txt"**,**"w"**);  
 **for**(**int** i=0 ; i<n ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 fprintf(fOutput,**"%d "**,matrix[i][j]);  
 }  
 fprintf(fOutput,**"\n"**);  
 }  
 **return** 0;  
}

# 43.

*/\*N positive integers are entered from the standard input and are stored in an array (N <100).  
All elements that meet the next criteria should be presented on the standard output in the same  
order that they have appeared on the standard input. The criteria is following: the most important  
digit (the digit on the first place) is odd and the least important digit is even. If, no such number  
was entered, you should print out "No such elements!" message.  
  
The implementation of checking the criteria should be performed in a separate  
function (recursive function for additional 5 points).\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>**#include **<math.h>  
  
int** countOfDig(**int** n){  
 **int** count=1;  
 **while**(n){  
 count\*=10;  
 n/=10;  
 }  
 count/=10;  
 **return** count;  
}  
**int** criteria(**int** n){  
 **int** d= countOfDig(n);  
 **int** fd=n/d;  
 **int** ld=n%10;  
 **return** fd%2!=0 && ld%2==0;  
}  
  
**int** main() {  
 **int** n;  
 scanf(**"%d"**,&n);  
 **int** array[100];  
 **for**(**int** i=0 ; i<n ; i++){  
 scanf(**"%d"**,&array[i]);  
 }  
 **int** flag=1;  
 **for**(**int** i=0 ; i<n ; i++){  
 **if**(criteria(array[i])){  
 printf(**"%d\n"**,array[i]);  
 flag=0;  
 }  
 }  
 **if**(flag){  
 printf(**"No such elements!"**);  
 }  
 **return** 0;  
}

# 44.

*/\*Read the dimensions M and N (not bigger than 100) and the values (whole numbers)  
of one matrix A\_ {MxN} from the standard input. Print out the index of the first  
column that has the smallest sum of all of its elements.\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>**#include **<math.h>  
  
  
int** main(){  
 **int** m,n;  
 scanf(**"%d%d"**,&m,&n);  
 **int** matrix[100][100];  
  
 **for**(**int** i=0 ; i<m ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 scanf(**"%d"**,&matrix[i][j]);  
 }  
 }  
 **int** sum,minSum,minIndex;  
 **int** flag=1;  
 **for**(**int** j=0 ; j<n ; j++){  
 sum=0;  
 **for**(**int** i=0 ; i<m ; i++){  
 sum+=matrix[i][j];  
 }  
 **if**(flag){  
 minSum=sum;  
 minIndex=j;  
 flag=0;  
 } **else if**(sum<minSum){  
 minSum=sum;  
 minIndex=j;  
 }  
 }  
 printf(**"%d"**,minIndex);  
 **return** 0;  
}

# 45.

*/\*A text file "input.txt" consist of digits and letters. The number of digits and letters  
in a row is limited to 80. Write a program that for every row will print out the letters  
concatenated next to each other first followed by the sum of all numbers that appear in that row.  
If there are no digits in the row, for the sum of the numbers you should print out 0.\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>**#include **<math.h>***//ne menuvaj!***void** wtf() {  
 FILE \*f = fopen(**"input.txt"**, **"w"**);  
 **char** c;  
 **while**((c = getchar()) != **'#'**) {  
 fputc(c, f);  
 }  
 fclose(f);  
}  
  
**int** isNumber(**char** \*str){  
 **int** sum=0;  
 **int** currentNumber=0;  
 **for**(**int** i=0 ; i< strlen(str) ; i++){  
 **if**(isdigit(str[i])){  
 currentNumber=10\*currentNumber+(str[i]-**'0'**);  
 } **else**{  
 sum+=currentNumber;  
 currentNumber=0;  
 }  
 }  
 **return** sum;  
}  
**void** letters(**char** \*str){  
 **for**(**int** i=0 ; i< strlen(str) ; i++){  
 **if**(isalpha(str[i])){  
 printf(**"%c"**,str[i]);  
 }  
 }  
}  
  
**int** main() {  
 wtf();  
 FILE \*f= fopen(**"input.txt"**,**"r"**);  
 **char** row[100];  
 **while** (fgets(row,**sizeof**(row),f)!=**NULL**){  
 letters(row);  
 **int** sum= isNumber(row);  
 printf(**"%d\n"**,sum);  
 }  
 fclose(f);  
 **return** 0;  
}

# 46.

E bame

# 47.

#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>**#include **<math.h>  
  
int** maxDN(**int** a, **int** b, **int** c, **int** d){  
 **int** max=a;  
 **if**(b>max){  
 max=b;  
 }  
 **if**(c>max){  
 max=c;  
 }  
 **if**(d>max){  
 max=d;  
 }  
 **return** max;  
}  
  
**int** main(){  
 **int** m,n;  
 scanf(**"%d%d"**,&m,&n);  
  
 **int** matrix[100][100];  
 **for**(**int** i=0 ; i<m ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 scanf(**"%d"**,&matrix[i][j]);  
 }  
 }  
 **int** transformed[100][100];  
 **for**(**int** i=0 ; i<m ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 transformed[i][j]=maxDN(matrix[i-1][j-1], matrix[i-1][j+1],matrix[i+1][j-1],matrix[i+1][j+1]);  
 }  
 }  
 **for**(**int** i=0 ; i<m ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 printf(**"%d "**,transformed[i][j]);  
 }  
 printf(**"\n"**);  
 }  
 **return** 0;  
}

# 48.

#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>**#include **<math.h>  
  
void** fivePercentages(**int** number, **int** percentage){  
 **if**(percentage>100){  
 **return**;  
 } **else**{  
 printf(**"%d%% = %.2f\n"**,percentage,number\*percentage/100.0);  
 fivePercentages(number,percentage+5);  
 }  
}  
  
**int** main(){  
 **int** number;  
 scanf(**"%d"**,&number);  
 fivePercentages(number,5);  
 **return** 0;  
}

# 49.

#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>**#include **<math.h>  
  
int** sumOfPairsLessThan(**int** n, **int** number){  
 **if**(number<100){  
 **if**(number<n){  
 **return** number;  
 } **else**{  
 **return** 0;  
 }  
 } **else**{  
 **int** pair=number%100;  
 **if**(pair<n){  
 **return** pair + sumOfPairsLessThan(n,number/100);  
 } **else**{  
 **return** sumOfPairsLessThan(n,number/100);  
 }  
 }  
}  
  
**int** main(){  
 **int** n,number;  
 scanf(**"%d%d"**,&number,&n);  
 printf(**"%d"**, sumOfPairsLessThan(n,number));  
 **return** 0;  
}

# 50.

E bame

# 51.

# 52.

# 