CSE 4304: Data Structure Lab

Lab 3

Solving problems related to Arrays and Sorting

Problem A

In many computer problems, it is necessary to permute data arrays. That is, the data in an array must be re-arranged in some specified order. One way to permute arbitrary data arrays is to specify the permutations with an index array to point out the position of the elements in the new array. Let x be an array that is to be permuted and let x' be the permuted array. Then, we have the relationship between x and x' that $x'_{pi} = x_i$.

Input:

The input begins with a single positive integer on a line by itself indicating the number of the test cases following, each of them as described below.

Each input set will contain two lines of numbers. The first line will be an index array p containing the integers 1...n, where n is the number of integers in the list. The numbers in the first line will have been permuted in some fashion. The first line takes input until a *'-1'* value is given indicating the end of the line. The second line will contain a list numbers in floating point format.

Output:

For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line.

The output for this program will be the list of floating-point numbers from the input set, ordered according to the permutation array from the input file. The output numbers must be printed one per line in the same format in which they each appeared in the input file.

Sample Input	Sample Output
2 3 1 2 -1 32.0 54.7 -2 5 3 6 2 1 4 -1 3.75 3.88 3.75 3.92 3.99 3.84	54.7 -2 32.0 3.99 3.92 3.88 3.84 3.75 3.75

Problem B

Xenia the beginner mathematician is a third-year student at elementary school. She is now learning the addition operation.

The teacher has written down the sum of multiple numbers. Pupils should calculate the sum. To make the calculation easier, the sum only contains numbers 1, 2 and 3. Still, that isn't enough for Xenia. She is only beginning to count, so she can calculate a sum only if the summands follow in non-decreasing order. For example, she can't calculate sum 1+3+2+1 but she can calculate sums 1+1+2 and 3+3.

You've got the sum that was written on the board. Rearrange the summans and print the sum in such a way that Xenia can calculate the sum.

Input:
The first line contains a non-empty string ***s*** — the sum Xenia needs to count. String ***s*** contains no spaces. It only contains digits and characters "+". Besides, string ***s*** is a correct sum of numbers 1, 2 and 3. String ***s*** is at most 100 characters long.

Output:
Print the new sum that Xenia can count.

Sample Input	Sample Output
3+2+1	1+2+3
1+1+3+1+3	1+1+1+3+3
2	2
1+2+2+2+1+3	1+1+2+2+2+3
3+3+3+3+3	3+3+3+3+3

Problem C

You are the king of Pensville where you have *2N* workers.
All workers will be grouped in association of size *2*, so a total of *N* associations have to be formed.
The building speed of the i^{th} worker is A_i.
To make an association, you pick up *2* workers. Let the minimum building speed between both workers be *x*, then the association has the resultant building speed *x*.
You have to print the maximum value possible of the sum of building speeds of *N* associations if you make the associations optimally.

Constraints

$1 \le N \le 5*10^4$
$1 \le A_i \le 10^4$

Input

First line contains an integer *N*, representing the number of associations to be made.
Next line contains *2N* space separated integers, denoting the building speeds of *2N* workers.

Output

Print the maximum value possible of the sum of building speeds of all the associations.

Sample Input_1
2
1 3 1 2
Sample output_1
3
[Hint: If you make an association using the first and third worker, and another using the second and fourth worker, each association will have *1* and *2* resultant building speed, which has a total of *3*.]

Sample Input_2
49
100 99 98 97 96 95 94 93 92 91 90 89 88 87 86 85 84 83 82 81 80 79 78 77 76 75 74 73 72 71 70 69 68 67 66 65 64 63 62 61 60 59 58 57 56 55 54 53 52 51 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3
Sample output_2
2499

Sample Input_3
1
1 1
Sample output_3
1

Problem D

One measure of “unsortedness” in a sequence is the number of pairs of entries that are out of order with respect to each other. For instance, in the letter sequence “DAABEC” this measure is 5, since D is greater than four letters to its right and E is greater than one letter to its right. This measure is called the number of inversions in the sequence. The sequence “AACEDGG” has only one inversion (E and D) --it is nearly sorted--while the sequence “ZWQM” has 6 inversions (it is as unsorted as can be--exactly the reverse of sorted).

You are responsible for cataloguing a sequence of DNA strings (sequences containing only the four letters A, C, G, and T). However, you want to catalog them, not in alphabetical order, but rather in order of “sortedness”, from ``most sorted'' to “least sorted”. All the strings are of the same length.

Input

The first line of the input is an integer M, then a blank line followed by M datasets. There is a blank line between datasets.

The first line of each dataset contains two integers: a positive integer *n* ($0 \leq n \leq 50$) giving the length of the strings; and a positive integer *m* ($0 \leq m \leq 100$) giving the number of strings. These are followed by *m* lines, each containing a string of length *n*.

Output

For each dataset, output the list of input strings, arranged from ``most sorted'' to ``least sorted''. If two or more strings are equally sorted, list them in the same order they are in the input file.
In the right of each string, it will show the degree of unsortedness of each string as well.

Print a blank line between consecutive test cases.

Sample Input

1
10 6
AACATGAAGG
TTTTGGCCAA
TTTGGCCAAA
GATCAGATTT
CCCGGGGGGA
ATCGATGCAT

Sample Output

CCCGGGGGGA 9
AACATGAAGG 10
GATCAGATTT 11
ATCGATGCAT 17
TTTTGGCCAA 36
TTTGGCCAAA 37

[Take the input from *‘in_d.txt’* file. Use *freopen(“in_txt”, “r”, stdlib);* to read the file.]