Homework: Java Collections Basics

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

Problem 1. Sort Array of Numbers

Write a program to **enter a number n** and **n integer numbers and sort and print them**. Keep the numbers in array of integers: **int[]**. Examples:

Input	Output
7 6 5 4 10 -3 120 4	-3 4 4 5 6 10 120
3 10 9 8	8 9 10
1 999	999

Problem 2. Sequences of Equal Strings

Write a program that **enters an array of strings** and finds in it **all sequences of equal elements**. The input strings are given as a single line, separated by a space. Examples:

Input	Output
hi yes yes yes bye	hi yes yes yes bye
SoftUni softUni softuni	SoftUni softUni softuni
1 1 2 2 3 3 4 4 5 5	1 1 2 2 3 3 4 4 5 5
a b b xxx c c c	a b b xxx c c c
hi hi hi hi hi	hi hi hi hi hi
hello	hello

Note: the count of the input strings is not explicitly specified, so you might need to read the first input line as a string and split it by a space.

Problem 3. Largest Sequence of Equal Strings

Write a program that **enters an array of strings** and finds in it **the largest sequence of equal elements**. If several sequences have the same longest length, print the **leftmost** of them. The input strings are given as a single line, separated by a space. Examples:

Input	Output
hi **yes yes yes** bye	yes yes yes
SoftUni softUni softuni	SoftUni
1 1 2 2 3 3 4 4 5 5	1 1
a b b xxx **c c c**	c c c
hi hi hi hi hi	hi hi hi hi hi
hello	hello

Problem 4. Longest Increasing Sequence

Write a program to **find all increasing sequences** inside an array of integers. The integers are given in a single line, separated by a space. Print the sequences in the order of their appearance in the input array, each at a single line. Separate the sequence elements by a space. Find also **the longest increasing sequence** and print it at the last line. If several sequences have the same longest length, print the leftmost of them. Examples:

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

Note: the count of the input numbers is not explicitly specified, so you might need to read the sequence as string, then split it by a space and finally parse the obtained tokens to take their integer values.

Problem 5. Count All Words

Write a program to **count the number of words** in given sentence. Use any non-letter character as word separator.

Examples:

Input	Output
Welcome to the Software University (SoftUni)!	6
I am coming...	3
It's OK, I'm in.	6
Java is a set of several computer software products and specifications from Oracle Corporation that provides a system for developing application software and deploying it in a cross-platform computing environment. Java is used in a wide variety of computing platforms from embedded devices and mobile phones on the low end, to enterprise servers and supercomputers on the high end.	60

Problem 6. Count Specified Word

Write a program to **find how many times a word appears in given text**. The text is given at the first input line. The target word is given at the second input line. The output is an integer number. Please ignore the character casing. Consider that any non-letter character is a word separator. Examples:

Input	Output
Welcome to the Software University (SoftUni)! Welcome to programming. welcome	2
I am coming... hello	0
It's OK, I'm in. i	1
Java is a set of several computer software products and specifications from Oracle Corporation that provides a system for developing application software and deploying it in a cross-platform computing environment. Java is used in a wide variety of computing platforms from embedded devices and mobile phones on the low end, to enterprise servers and supercomputers on the high end. is	2

Problem 7. Count Substring Occurrences

Write a program to **find how many times given string appears in given text as substring**. The text is given at the first input line. The search string is given at the second input line. The output is an integer number. Please ignore the character casing. Examples:

Input	Output
Welcome to the Software University (SoftUni)! **Wel**come to programming. Programming is **wel**lness for developers, said Max**wel**l. wel	4
aaaaaa aa	5
ababa caba aba	3

Welcome to SoftUni Java	0

Problem 8. Extract Emails

Write a program to **extract all email addresses from given text**. The text comes at the first input line. Print the emails in the output, each at a separate line. Emails are considered to be in format **<user>@<host>**, where:

- **<user>** is a sequence of letters and digits, where '**.**', '**-**' and '**_**' can appear between them. Examples of valid users: "**stephan**", "**mike03**", "**s.johnson**", "**st_steward**", "**softuni-bulgaria**", "**12345**". Examples of invalid users: "**--123**", ".....", "**nakov_-**", "**_steve**", "**.info**".
- **<host>** is a sequence of at least two words, separated by dots '**.**'. Each word is sequence of letters and can have hyphens '**-**' between the letters. Examples of hosts: "**softuni.bg**", "**software-university.com**", "**intoprogramming.info**", "**mail.softuni.org**". Examples of invalid hosts: "**helloworld**", "**.unknown.soft.**", "**invalid-host-**", "**invalid-**".
- Example of **valid emails**: info@softuni-bulgaria.org, kiki@hotmail.co.uk, no-reply@github.com, s.peterson@mail.uu.net, info-bg@software-university.software.academy.

Examples:

Input	Output
Please contact us at: support@github.com.	support@github.com
Just send email to s.miller@mit.edu and j.hopking@york.ac.uk for more information.	s.miller@mit.edu j.hopking@york.ac.uk
Many users @ SoftUni confuse email addresses. We @ Softuni.BG provide high-quality training @ home or @ class. -- steve.parker@softuni.de.	steve.parker@softuni.de

Problem 9. Combine Lists of Letters

Write a program that **reads two lists of letters l1 and l2 and combines them**: appends all letters **c** from **l2** to the end of **l1**, but only when **c** does not appear in **l1**. Print the obtained combined list. All lists are given as sequence of letters separated by a single space, each at a separate line. Use **`ArrayList<Character>`** of chars to keep the input and output lists. Examples:

Input	Output
h e l l o l o w	h e l l o w
a b c d x y z	a b c d x y z
a b a b a b a	a b a
w e l c o m e t o s o f t u n i j a v a p r o g r a m m i n g	w e l c o m e t o s o f t u n i j a v a p r g r a g

Problem 10. Extract All Unique Words

At the first line at the console you are given a piece of **text**. **Extract all words from it** and print them in **alphabetical order**. Consider each non-letter character as word separator. Take the repeating words only once. Ignore the character casing. Print the result words in a single line, separated by spaces. Examples:

Input	Output
Welcome to SoftUni. Welcome to Java.	java to softuni welcome
What is better: Java SE or Java EE?	better ee is java or se what
hello	hello

Problem 11. Most Frequent Word

Write a program to **find the most frequent word** in a text and print it, as well as **how many times it appears** in format "**word -> count**". Consider any non-letter character as a word separator. Ignore the character casing. If several words have the same maximal frequency, print all of them in alphabetical order. Examples:

Input	Output
in the middle of the night	the -> 2 times
Welcome to SoftUni. Welcome to Java. Welcome everyone.	welcome -> 3 times
Hello my friend, hello my darling. Come on, come here. Welcome, welcome darling.	come -> 2 times darling -> 2 times hello -> 2 times my -> 2 times welcome -> 2 times

Problem 12. Cards Frequencies

We are given a sequence of **N playing cards** from a standard deck. The input consists of several cards (face + suit), separated by a space. Write a program to calculate and print at the console the frequency of each card face in format "**card_face -> frequency**". The frequency is calculated by the formula **appearances / N** and is expressed in percentages with exactly 2 digits after the decimal point. The card faces with their frequency should be printed in the order of the card face's first appearance in the input. The same card can appear multiple times in the input, but it's face should be listed only once in the output. Examples:

Input	Output
8♥ 2♣ 4♦ 10♦ J♥ A♠ K♦ 10♥ K♠ K♦	8 -> 10.00% 2 -> 10.00% 4 -> 10.00% 10 -> 20.00% J -> 10.00% A -> 10.00% K -> 30.00%
J♥ 2♣ 2♦ 2♥ 2♦ 2♠ 2♦ J♥ 2♠	J -> 22.22% 2 -> 77.78%
10♣ 10♥	10 -> 100.00%

Problem 13. ** Web Crawler

A [web crawler](#) (web spider) is a program that crawls the web starting from a certain URL and passing to all its internal and external hyperlinks recursively until stopped, until reached something or up to certain predefined depth. Your task is to write a web crawler that starts from certain URL and crawls all URLs recursively up to **n** levels or until stopped. The number **n** (e.g. 2) and the starting URL are given as constants. All non-html documents should be ignored (e.g. images, CSS styles and scripts). Write all crawled URLs in a text file "**crawled-urls.txt**". Don't crawl the same URL more than once. You may use any external library: HTML parsers, crawler engines, etc.

Exam problems.** - Java Basics Exam 1st June 2014

All of the problems below are given from the previous Java Basics exams. **You are not obligated** to submit any of them in your homework. We highly recommend you to try solving some or all of them so you can be well prepared for the upcoming exam. You need to learn how to use conditional statements, loops, arrays and other things (learn in internet how or read those chapters in the book "Fundamentals of computer programming with Java"). If you still find those problems too hard for solving it's very useful to **check** and **understand** the solutions. You can download all solutions and tests for this variant here or check all previous exams (scroll down to the bottom of the page). You can also test your solutions in our automated judge system to see if you pass all tests.

Problem 14* – Stuck Numbers

You are given **n numbers**. Write a program to find among these numbers all sets of 4 numbers {**a**, **b**, **c**, **d**}, such that **a**|**b** == **c**|**d**, where **a** ≠ **b** ≠ **c** ≠ **d**. Assume that "**a**|**b**" means to append the number **b** after **a**. We call these numbers {**a**, **b**, **c**, **d**} **stuck numbers**: if we append **a** and **b**, we get the same result like if we append **c** and **d**. Note that the numbers **a**, **b**, **c** and **d** should be distinct (**a** ≠ **b** ≠ **c** ≠ **d**).

Input

The input comes from the console. The first line holds the **count n**. The next line holds **n integer numbers**, separated by a space. The input numbers will be **distinct** (no duplicates are allowed).

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

Output

Print at the console all **stuck numbers** {**a**, **b**, **c**, **d**} found in the input sequence in format "**a**|**b**==**c**|**d**" (without any spaces), each at a separate line. The **order** of the output lines **is not important**. Print "**No**" in case no stuck numbers exist among the input sequence of numbers.

Constraints

- The **count n** will be an integer number in the range [1…50].
- The input **numbers** will be **distinct** integers in the range [0…9999].
- Time limit: 0.5 sec. Memory limit: 16 MB.

Examples

Input	Output
5 2 51 1 75 25	2|51==25|1 25|1==2|51

Input	Output
7 2 22 23 32 322 222 5	2|322==23|22 23|22==2|322 32|22==322|2 32|222==322|22 322|2==32|22 322|22==32|222

Input	Output
3 5 1 20	No

Problem 15** – Sum Cards

Nakov is keen card player and he is now learning a new game. The game uses a **standard deck of 52 cards**. The card faces are: **2**, **3**, **4**, **5**, **6**, **7**, **8**, **9**, **10**, **J**, **Q**, **K** and **A**. The cards suits are denoted by the letters **S** (spades), **H** (hearts), **D** (diamonds) and **C** (clubs). The player is given a hand of cards and he needs to **count their sum**. Card **values** are the

following: 2 -> 2, 3 -> 3, 4 -> 4, 5 -> 5, 6 -> 6, 7 -> 7, 8 -> 8, 9 -> 9, 10 -> 10, J -> 12, Q -> 13, K -> 14, A -> 15 (the card suits are ignored). When two or more cards of the same face **come sequentially**, their values are **counted twice**.

For example, the hand "**2C 2H 2D AS 10H 10C 2S KD**" has value (2 + 2 + 2) * 2 + 15 + (10 + 10) * 2 + 2 + 14 = **83**.

Write a program that **takes a hand of cards and counts their sum**.

Input

The input comes from the console as a **single line, holding the hand of cards**. Cards are separated by a space.

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

Output

Print at the console a single number: the **value of the hand**.

Constraints

- The **count** the cards will be in the range [1…99].
- **Card faces** will be one of the following values: [**2**, **3**, **4**, **5**, **6**, **7**, **8**, **9**, **10**, **J**, **Q**, **K**, **A**].
- **Card suits** will be one of the following values: [**S**, **H**, **D**, **C**].
- Time limit: 0.3 sec. Memory limit: 16 MB.

Examples

Input	Output
2C 2H 2D AS 10H 10C 2S KD	83
AS KH 10C	39
2S 2C 2D 2H	16
AS 10C KS KH KD 9H JH QS 3H QD QH 8S 10D 10S 7C JD	265

Problem 16** – Simple Expression

You are given an **arithmetic expression**, consisting of **positive numbers** and '**+**' and '**-**' between them. Write a program to **calculate the value of the expression**.

Input

The input comes from the console. It consists of a **single line holding the arithmetic expression**. It consists of positive numbers and '**+**' and '**-**' between them. Anywhere around the numbers spaces could appear.

Output

Print the **value of the input expression** at the console with a **precision of 7 decimal digits**. Don't use scientific notation in the output! This means that if the output is **0.15**, it will be considered correct if it is printed as **0.1500**, as **.15** or as **0.150**, but **not** as **1.5e-1**.

Constraints

- The **input numbers** will be standard floating-point values in the range [-10^{10}…10^{10}] with no more than 10 digits. Use the "**.**" symbol as decimal separator. The scientific notation (e.g. **1.5e-12**) may not be used in the input numbers.

- The **output numbers** will be standard floating-point values in the range [-10^{15}…10^{15}] with no more than 30 digits. Use the "**.**" symbol as decimal separator. The scientific notation (e.g. **1.5e-12**) may not be used in the output numbers. Optionally, the output numbers can be rounded up to 7 digits after the decimal point.
- The **count** of the input numbers will be less than 1000.
- Time limit: 0.3 sec. Memory limit: 16 MB.

Examples

Input	Output	Comments
`5 -33 + 12 - 55- 1 - 2+6`	`-68`	`-68.0 is also correct`
`1.5 + 2.5`	`4.0`	`4 and 4.00 are also correct`
`0.05+0.01 - 1`	`-0.94`	`-0.9400000 is also corect`
`1 + 2`	`3`	`3.0 is also corect`
`9876543210 + 0.987654321`	`9876543210.987654321`	`9876543210.9876543 is also correct`

Problem 17*** – Logs Aggregator

You are given a sequence of access logs in format **<IP> <user> <duration>**. For example:

- 192.168.0.11 peter 33
- 10.10.17.33 alex 12
- 10.10.17.35 peter 30
- 10.10.17.34 peter 120
- 10.10.17.34 peter 120
- 212.50.118.81 alex 46
- 212.50.118.81 alex 4

Write a program to aggregate the logs data and print **for each user** the **total duration** of his sessions and a **list of unique IP addresses** in format "**<user>: <duration> [<IP$_1$>, <IP$_2$>, …]**". Order the **users alphabetically**. Order the **IPs alphabetically**. In our example, the output should be the following:

- alex: 62 [10.10.17.33, 212.50.118.81]
- peter: 303 [10.10.17.34, 10.10.17.35, 192.168.0.11]

Input

The input comes from the console. At the first line a number **n** stays which says how many log lines will follow. Each of the next n lines holds a log information in format **<IP> <user> <duration>**. The input data will always be valid and in the format described. There is no need to check it explicitly.

Output

Print **one line for each user** (order users alphabetically). For each user print its sum of durations and all of his sessions' IPs, ordered alphabetically in format **<user>: <duration> [<IP$_1$>, <IP$_2$>, …]**. Remove any duplicated values in the IP addresses and order them alphabetically (like we order strings).

Constraints

- The **count** of the order lines **n** is in the range [1…1000].
- The **<IP>** is a standard IP address in format **a.b.c.d** where **a**, **b**, **c** and **d** are integers in the range [0…255].
- The **<user>** consists of only of **Latin characters**, with length of [1…20].

- The **<duration>** is an integer number in the range [1…1000].
- Time limit: 0.3 sec. Memory limit: 16 MB.

Examples

Input	Output
7 192.168.0.11 peter 33 10.10.17.33 alex 12 10.10.17.35 peter 30 10.10.17.34 peter 120 10.10.17.34 peter 120 212.50.118.81 alex 46 212.50.118.81 alex 4	alex: 62 [10.10.17.33, 212.50.118.81] peter: 303 [10.10.17.34, 10.10.17.35, 192.168.0.11]
2 84.238.140.178 nakov 25 84.238.140.178 nakov 35	nakov: 60 [84.238.140.178]