Sirma

e: office@sirma.bg
m: +359 2 9768310
a: 135 Tsarigradsko Shosse, blvd.
1784 Sofia, Bulgaria

Sets & Maps - Exercises

1. Count Real Numbers

Write a program that counts the occurrence of real numbers. The input is a single line with real numbers separated by a space. Print the numbers in order of appearance. All numbers must be formatted to one digit after the decimal point.

Input	Output
-2.5 4 3 -2.5 -5.5 4 3 3 -2.5 3	**-2.5 -> 3** **4.0 -> 2** **3.0 -> 4** **-5.5 -> 1**

e: office@sirma.bg
m: +359 2 9768310
a: 135 Tsarigradsko Shosse, blvd.
1784 Sofia, Bulgaria

2.3 4.5 4.5 5.5 5.5 2.3 3.0 3.0 4.5 4.5 3.0 3.0 4.0 3.0 5.5 3.0 2.3 5.5 4.5 3.0	**2.3 -> 3** **4.5 -> 5** **5.5 -> 4** **3.0 -> 7** **1.0 -> 1**

2. Average Students Grades

Write a program, which reads the name of a student and their grades and adds them to the student record, then prints grades along with their average grade – ordered the output by the students' names.

- Use a TreeMap (String -> ArrayList<Double>) to ensure correct order.
- Check if the name exists before adding the grade. If it doesn't, add it to the map.

Input	Output
7 **Stephan 5.20** **Maria 5.50** **Stephan 3.20** **Maria 2.50** **Alex 2.00** **Maria 3.46** **Alex 3.00**	**Alex -> 2.00 3.00 (avg: 2.50)** **Maria -> 5.50 2.50 3.46 (avg: 3.82)** **Stephan -> 5.20 3.20 (avg: 4.20)**
4 **Alex 4.50** **Peter 3.00** **Alex 5.00** **Peter 3.66**	**Alex -> 4.50 5.00 (avg: 4.75)** **Peter -> 3.00 3.66 (avg: 3.33)**
5 **George 6.00** **George 5.50** **George 6.00** **Alex 4.40**	**Alex -> 4.40 (avg: 4.40)** **George -> 6.00 5.50 6.00 (avg: 5.83)** **Peter -> 3.30 (avg: 3.30)**

e: office@sirma.bg
m: +359 2 9768310
a: 135 Tsarigradsko Shosse, blvd.
1784 Sofia, Bulgaria

Peter 3.30 **Peter 4.50**	

3. Count Symbols

Write a program that reads some text from the console and counts the occurrences of each character in it. Print the results in alphabetical (lexicographical) order.

Input	Output
Java rocks	**: 1** **J: 1** **a: 2** **c: 1** **k: 1** **o: 1** **r: 1** **s: 1** **v: 1**

4. Phonebook

Write a program that receives some info from the console about people and their phone numbers.

Each entry should have just one name and one number. If you receive a name that already exists in the phonebook, simply update its number.

After filling this simple phonebook, upon receiving the command "search", your program should be able to perform a search of contact by name and print details in the format "{name} -> {number}". In case the contact isn't found, print "Contact {name} not found.".

Input	Output
John-0888888888 **search** **Mary** **John** **stop**	**Contact Mary not found.** **John -> 0888888888**

e: office@sirma.bg
m: +359 2 9768310
a: 135 Tsarigradsko Shosse, blvd.
1784 Sofia, Bulgaria

John-0888888888	**Sam -> 0047123123123**
Peter-0040111111000	**Contact sam does not exist.**
George-0049112233	**Sam -> 0047111111111**
Sam-0047123123123	**Contact PeTeR does not exist.**
search	**Peter -> 0040111111000**
Sam	
sam	
Sam-00471111211111	
Sam	
PeTeR	
Peter	
stop	

5. Hands Of Cards

You are given a sequence of people and what cards he draws from the deck for every person. The input will be separate lines in the format:

"{personName}: {PT, PT, PT,… PT}"

Where P (2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K, A) is the power of the card and T (S, H, D, C) is the type. The input ends when a "JOKER" is drawn.

A single person cannot have more than one card with the same power and type. If he draws such a card, he discards it. The people are playing with multiple decks. Each card has a value that is calculated by the power multiplied by the type. Powers 2 to 10 have the same value, and J to A is 11 to 14. Types are mapped to multipliers the following way (S -> 4, H-> 3, D -> 2, C -> 1).

Finally, print out the total value each player has in his hand in the format:

"{personName}: {value}"

Input	Output
Peter: 2C, 4H, 9H, AS, QS	**Peter: 167**
Marry: 3H, 10S, JC, KD, 5S, 10S	**Marry: 175**
Alex: QH, QC, QS, QD	**Alex: 197**
Marry: 6H, 7S, KC, KD, 5S, 10C	
Alex: QH, QC, JS, JD, JC	

e: office@sirma.bg
m: +359 2 9768310
a: 135 Tsarigradsko Shosse, blvd.
1784 Sofia, Bulgaria

Peter: JD, JD, JD, JD, JD, JD **JOKER**	
JJ: JD, JD, JD, JD **JOKER**	**JJ: 22**

6. Population Counter

You get raw data for a given city, and you need to aggregate it.

On each input line, you'll be given data in the format: "city|country|population".

Aggregate the data by country and by city and print it on the console. For each country, print its total population and on separate lines the data for each of its cities. Countries should be ordered by their total population in descending order, and within each country, the cities should be ordered by the same criterion. If two countries/cities have the same population, keep them in the order in which they were entered. Check out the examples. Follow the output format strictly!

- The input data should be read from the console.
- It consists of a variable number of lines and ends when the command "report" is received.
- The input data will always be valid and in the format described. There is no need to check it explicitly.

Input	Output
Sofia|Bulgaria|1000000 **report**	**Bulgaria (total population: 1000000)** **=>Sofia: 1000000**
Sofia|Bulgaria|1 **Varna|Bulgaria|2** **London|UK|4** **Rome|Italy|3** **report**	**UK (total population: 4)** **=>London: 4** **Bulgaria (total population: 3)** **=>Varna: 2** **=>Sofia: 1** **Italy (total population: 3)** **=>Rome: 3**

e: office@sirma.bg
m: +359 2 9768310
a: 135 Tsarigradsko Shosse, blvd.
1784 Sofia, Bulgaria

7. Word Synonyms

Write a program that keeps a dictionary with synonyms. The key to the map will be the word. The value will be a list of all the synonyms of that word. You will be given number n. On the next 2 * n lines, you will be given the word and a synonym each on a separate line like this:

- {word}
- {synonym}

If you get the same word for the second time, just add the new synonym to the list.

Print the words in the following format:

{word} - {synonym1, synonym2… synonymN}

- Use LinkedHashMap (String -> ArrayList<String>) to keep track of all words.

Input	Output
3 **cute** **adorable** **cute** **charming** **smart** **clever**	**cute - adorable, charming** **smart - clever**
2 **task** **problem** **task** **assignment**	**task – problem, assignment**

8. Odd Occurrences

Write a program that extracts from a given sequence of words all elements that are present in it an odd number of times (case-insensitive).

- Words are given in a single line, space separated.
- Print the result elements in lowercase in their order of appearance.

Input	Output

Java PHP PHP JAVA C java	**java, c**
3 5 5 hi pi HO Hi 5 ho 3 hi pi	**5, hi**
a a A SQL xx a xx a A a XX c	**a, sql, xx, c**

9. Word Filter

Read an array of strings and take only words whose length is even. Print each word on a new line.

Input	Output
kiwi orange banana apple	**kiwi** **orange** **banana**
pizza cake pasta chips	**cake**

10. Cities by Continent and Country

Write a program to read continents, countries, and their cities, put them on a nested map, and print them in the order of their first appearance.

Input	Output
9 **Europe Bulgaria Sofia** **Asia China Beijing** **Asia Japan Tokyo** **Europe Poland Warsaw** **Europe Germany Berlin** **Europe Poland Poznan** **Europe Bulgaria Plovdiv** **Africa Nigeria Abuja** **Asia China Shanghai**	**Europe:** **Bulgaria -> Sofia, Plovdiv** **Poland -> Warsaw, Poznan** **Germany -> Berlin** **Asia:** **China -> Beijing, Shanghai** **Japan -> Tokyo** **Africa:** **Nigeria -> Abuja**
3 **Europe Germany Berlin** **Europe Bulgaria Varna**	**Europe:** **Germany -> Berlin** **Bulgaria -> Varna**

e: office@sirma.bg
m: +359 2 9768310
a: 135 Tsarigradsko Shosse, blvd.
1784 Sofia, Bulgaria

Africa Egypt Cairo	**Africa:** **Egypt -> Cairo**
8 **Africa Somalia Mogadishu** **Asia India Mumbai** **Asia India Delhi** **Europe France Paris** **Asia India Nagpur** **Europe Germany Hamburg** **Europe Poland Gdansk** **Europe Germany Danzig**	**Africa:** **Somalia -> Mogadishu** **Asia:** **India -> Mumbai, Delhi, Nagpur** **Europe:** **France -> Paris** **Germany -> Hamburg, Danzig** **Poland -> Gdansk**

- **Use a nested Map (String -> (Map -> ArrayList<String>)) .**
- **Check if the continent exists before adding the country. If it doesn't, add it to the dictionary.**
- **Check if the country exists before adding the city. If it doesn't, add it to the dictionary.**

11. Largest 3 Numbers

Read a list of integers and print the largest 3 of them. If there are less than 3, print all of them.

Input	Output
10 30 15 20 50 5	**50 30 20**
20 30	**30 20**

12. Count Chars in a String

Write a program that counts all characters in a string except space (' ').

Print all occurrences in the following format:

"{char} -> {occurrences}"

Examples

Input	Output
text	**t -> 2**

	e -> 1 **x -> 1**
text text text	**t -> 6** **e -> 3** **x -> 3**

13. Parking System

Write a program that validates cars for parking system. Users can register to park and unregister to leave.

The program receives 2 commands:

- "Register {username} {licensePlateNumber}":
  - The system only supports one car per user now, so if a user tries to register another license plate using the same username, the system should print:
    "ERROR: already registered with plate number {licensePlateNumber}"
  - If the checks pass successfully, the plate can be registered, so the system should print:
    "{username} registered {licensePlateNumber} successfully."
- "Unregister {username}":
  - If the user is not present in the database, the system should print:
    "ERROR: user {username} not found."
  - If the check passes successfully, the system should print:
    "{username} unregistered successfully."

After you execute all the commands, print the currently registered users and their license plates in the format:

- "{username} => {licensePlateNumber}"
- First line: n - number of commands – integer.
- Next n lines: commands in one of two possible formats:
  - Register: "register {username} {licensePlateNumber}"
  - Unregister: "unregister {username}"

Input	Output
5 **Register John CS1234JS** **Register George JAVA123S**	**John registered CS1234JS successfully.** **George registered JAVA123S successfully.** **Andy registered AB4142CD successfully.**

e: office@sirma.bg
m: +359 2 9768310
a: 135 Tsarigradsko Shosse, blvd.
1784 Sofia, Bulgaria

Register Andy AB4142CD **Register Jessica VR1223EE** **Unregister Andy**	**Jessica registered VR1223EE successfully.** **Andy unregistered successfully.** **John => CS1234JS** **George => JAVA123S** **Jessica => VR1223EE**
4 **Register John AA1234BB** **Register John AA1234BB** **Register Linda AA0000BB** **Unregister Jony**	**John registered AA1234BB successfully.** **ERROR: already registered with plate number AA1234BB** **Linda registered AA0000BB successfully.** **John unregistered successfully.** **Linda => AA0000BB**
6 **Register Jacob MM1111XX** **Register Anthony AB1111XX** **Unregister Jacob** **Register Joshua DD1111XX** **Unregister Lily** **Register Samantha AA0000BB**	**Jacob registered MM1111XX successfully.** **Anthony registered AB1111XX successfully.** **Jacob unregistered successfully.** **Joshua registered DD1111XX successfully.** **ERROR: user Lily not found** **Samantha registered AA0000BB successfully.** **Anthony => AB1111XX** **Joshua => DD1111XX** **Samantha => AA0000BB**

14. Student Academy

Write a program that keeps the information about students and their grades.

On the first line, you will receive number n. After that, you will receive n pair of rows. First, you will receive the student's name, after that, you will receive his grade. Check if the student already exists and if not - add him. Keep track of all grades for each student.

When you finish reading data, keep students with an average grade higher or equal to 4.50.

Print the students and their average grade in the format:

"{name} –> {averageGrade}"

e: office@sirma.bg
m: +359 2 9768310
a: 135 Tsarigradsko Shosse, blvd.
1784 Sofia, Bulgaria

Format the average grade to the 2nd decimal place.

Input	Output		Input	Output
5 John 5.5 John 4.5 Alice 6 Alice 3 George 5	John -> 5.00 Alice -> 4.50 George -> 5.00		5 Petra 3.5 Petra 4 Rob 5.5 Christian 5 Robert 6	Rob -> 5.50 Christian -> 5.00 Robert -> 6.00

15. Company Users

Write a program which keeps information about companies and their employees.

You will receive company names and an employees' id until you receive the "End" command. Add each employee to the given company. Keep in mind that a company cannot have two employees with the same id.

Print the company name and each employee's id in the following format:

"{company_name}

-- {id1}

-- {id2}

…

-- {idN}"

- Until you receive "End", the input come in the format: "{companyName} -> {employeeId}".

Input	Output
Sirma -> A12345 Sirma -> B12345 Microsoft -> C12345	Sirma -- A12345 -- B12345

e: office@sirma.bg
m: +359 2 9768310
a: 135 Tsarigradsko Shosse, blvd.
1784 Sofia, Bulgaria

HP -> B12345 **End**	**Microsoft** **-- C12345** **HP** **-- B12345**
Sirma -> A12345 **Sirma -> C12344** **Lenovo -> X23456** **Sirma -> A12345** **Movement -> D11111** **End**	**Sirma** **-- A12345** **-- C12344** **Lenovo** **-- X23456** **Movement** **-- D11111**

16. Parking Lot

Write a program that:

- Records **car numbers** for every car that enters the **parking lot.**
- Removes **car number** when the car is out.

When the parking lot is empty, print "**Parking Lot is Empty**".

The input will be a string in the format "**{direction, carNumber}**".

The input ends with the string "**END**".

Print the output with all car numbers which are in the parking lot.

Input	Output
IN, CA2844AA **IN, CA1234TA** **OUT, CA2844AA** **IN, CA9999TT** **IN, CA2866HI** **OUT, CA1234TA** **IN, CA2844AA** **OUT, CA2866HI** **IN, CA9876HH**	CA9999TT CA2844AA CA9876HH CA2822UU

IN, CA2822UU **END**	
IN, CA2844AA **IN, CA1234TA** **OUT, CA2844AA** **OUT, CA1234TA** **END**	Parking Lot is Empty

17. Party List

Write a program that tracks the guests invited to a party. There are two types of guests: **VIP** and **regular**.
When a guest comes, you must check if the guest **exists** on any of the two reservation lists. All reservation numbers will be **8 characters long. VIP** numbers start with a **digit.**

You will receive is "**PARTY**" - the party is on, and guests are coming. The second is "**END**" - then the party is over, and no more guests will come.

The output shows all guests who didn't come to the party (**VIP** must be first).

Input	Output	Input	Output
7IK9Yo0h **9NoBUajQ** **Ce8vwPmE** **tSzE5t0p** **PARTY** **9NoBUajQ** **Ce8vwPmE** **END**	2 7IK9Yo0h tSzE5t0p	**m8rfQBvl** **fc1oZCE0** **UgffRkOn** **7ugX7bm0** **9CQBGUeJ** **2FQZT3uC** **dziNz78I** **mdSGyQCJ** **LjcVpmDL** **fPXNHpm1** **HTTbwRmM** **B5yTkMQi** **8N0FThqG** **xys2FYzn**	2 MDzcM9ZK xys2FYzn

		MDzcM9ZK **PARTY** **2FQZT3uC** **dziNz78I** **mdSGyQCJ** **LjcVpmDL** **fPXNHpm1** **HTTbwRmM** **B5yTkMQi** **8N0FThqG** **m8rfQBvl** **fc1oZCE0** **UgffRkOn** **7ugX7bm0** **9CQBGUeJ** **END**	

18. War Game

Read 20 cards for both players, separated with " " (single space).

- Every player can hold only **unique** cards.

Each Round, both players get the **top card** from their deck. The player with the bigger card gets both cards and adds them to the **bottom** of his deck.

The game ends after **50 rounds** or if any player **loses all** his cards.

- Output must be "**First player wins!**", "**Second player wins!**" or "**Draw!**".

Input	Output
26 58 16 92 44 65 65 77 57 23 71 57 7 52 85 44 32 70 38 23 **43 95 33 51 62 93 57 55 0 31 32 95 68 34 30 51 37 32 11 97**	Second player wins!
74 78 82 42 19 39 29 69 20 42 31 77 57 36 76 26 4 9 83 42 **15 43 80 71 22 88 78 35 28 30 46 41 76 51 76 18 14 52 47 38**	First player wins!

e: office@sirma.bg
m: +359 2 9768310
a: 135 Tsarigradsko Shosse, blvd.
1784 Sofia, Bulgaria

19. Unique Usernames

Write a simple program that reads usernames from the console and keeps a collection with only the unique ones. Print the collection on the console in order of insertion:

Input	Output
5 **Hello** **Hello** **World** **Hello** **Greetings**	Hello World Greetings
10 **Peter** **Mary** **Peter** **George** **Stephen** **Mary** **Alex** **Peter** **Stephen** **George**	Peter Mary George Stephen Alex

20. Sets of Elements

On the first line, you are given the length of two sets, **N** and **M**. On the next **N + M** lines, there are **N** numbers that are in the **first** set and **M** numbers that are in the **second** one. Find all non-repeating element that appears in both, and print them in the same order at the console:

Set with length N = 4: {1, **2**, **6**, 7}

Set with length M = 3: {**2**, 4, **6**}

Set that contains all repeating elements -> {**2**, **6**}

Examples

e: office@sirma.bg
m: +359 2 9768310
a: 135 Tsarigradsko Shosse, blvd.
1784 Sofia, Bulgaria

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

21. Periodic Table

You are given several chemical compounds. You need to keep track of all chemical elements used in the compounds and at the end, print all unique ones in ascending order:

Examples

Input	Output
4 **He O** **Ni O He** **Ee** **Ni**	Ee He Ni O
3 **Ge Cl O Ni** **Na Mo Tc** **O Ni**	Cl Ge Mo Na Ni O Tc