# 09. Exercises on Associative Containers, Classes and Objects

# Problem 1 – What are the Odds

Write a program that reads a line of words (sequences of lowercase English letters separated by single spaces) and prints two lines – the first containing all words appearing an odd number of times, sorted alphabetically, and the second containing all words appearing an even number of times, sorted alphabetically.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| java cpp csharp php php java c java cpp | c csharp java  cpp php |

# Problem 2 – In Range

You are given a line of integer numbers, followed by another line containing exactly two numbers – the start (inclusive) and end (exclusive) of a range (start will always be less than end). Write a program that prints all numbers from the first line that fall into the range [start, end), in increasing order, without duplicates.

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1 42 13 1 13 10 9 7 4 105  7 12 | 7 9 10 |
| 1 4 7 9 10 13 42 105 |  |
| 1 42 13 1 13 10 9 4 105  7 13 | 9 10 |

# Problem 3 – Matches

You are given two rows of integer numbers. Print a single line containing all the numbers that appear both in the first and the second row (without repetitions), sorted in increasing order. If there are no numbers that appear in both rows, print no matches.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1 42 13 1 13 10 9 7 4 105  7 7 19 1 106 42 -9 1 1 1 1 1 1 | 1 7 42 |
| 1  2 | no matches |
| 42  42 42 | 42 |

# Problem 4 – Letters

You are given a text in English. Let’s define a word as any sequence of alphabetical characters. Each of those characters we’ll call a letter, but we will consider the uppercase and lowercase variant of a character in a word as the **same** letter.

Write a program which reads the text (a single line on the console) and then reads lines, each containing a single letter, until a line containing a '.' (dot) is entered. For each of those lines, print all words that contain the letter, ordered alphabetically (capitals letters before lowercase letters), without duplicates – if no words contain that letter, print "---" (three dashes)

### Examples

|  |
| --- |
| **Input (NOTE: the *italic* text is on a single line)** |
| *You are given a text in English. Let’s define a word as any sequence of alphabetical characters. Each of those characters we’ll call a letter, but we will consider the uppercase and lowercase variant of a character in a word as the* ***same*** *letter.*  a  Y  h  . |

|  |
| --- |
| **Output** |
| Each a alphabetical and any are as call character characters lowercase same uppercase variant  You any  Each English alphabetical characters the those |

# Problem 5 – Matching Locations

Write a program that reads **names** of places and their geographical **coordinates** in the format name,latitude,longitude (where latitude and longitude are floating-point numbers). No two locations will have the same name. Some locations may have the same **coordinates**.

After all locations are entered, a single line containing the '.' (dot) character will be entered.

After that, queries will be entered – the queries will either contain a name of a location, or latitude and longitude coordinates (entered as two floating point numbers separated by a single space). Print all locations that match the query in the same format that they were entered.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Sofia,42.70,23.33  New York,40.6976701,-74.2598732  SoftUni,42.70,23.33  .  Sofia  40.6976701 -74.2598732  42.70 23.33 | Sofia, 42.70,23.33  New York, 40.6976701,-74.2598732  Sofia, 42.70,23.33  SoftUni,42.70,23.33 |

# Task 6 – Astronomical Objects

You are tasked with creating a console application which will store and display information about astronomical objects. Each object has a name of a home solar system, a position in the system, a mass in kg, a radius in meters, and a type (star, rocky planet, gas giant, unknown) and a nickname. Only some astronomical objects have a nickname. The name of the home solar system of an object can change (the solar system could be renamed), and the type can ONLY change from unknown to star, rocky planet, or gas giant. Objects of type star always have their position in the system set to 1. The designation of an astronomical object has the format:

home solar system – position in the system (nickname)

Where (nickname) is omitted if the object has no nickname. E.g. the sun of the Cancri system will have a designation Cancri-1, while the 8th planet named “Steel World” will have a designation of Cancri-9 (Steel World).

Write a program, which allows the user to create info about astronomical objects as well as search for all planets of a star system (by typing the name of the system) or of a specific object of a system (by typing the name and position of the object). When displaying info about objects the user searched for, use the following format:

designation { mass: *mass in kg*, radius: *radius in meters* }

For example, if Steel World had a mass of 5.972e+24 kg and a radius of 6 137 000 meters, displaying it to the user would look like:   
Cancri-9 (Steel World) {mass: 5.972e+24, radius: 6137000} (note: don’t worry about the exact format of the numbers, just show the numbers however cout decides to print them).

Creating objects is up to you – just make sure a user can add the info of any type of astronomical object.

The program should store the info in a text file (each time an object is created) and should load the info each time it starts (so that users can look-up objects they created previously).

Make sure you create the proper classes, constructors, access modifiers and methods for the above task. You should submit your program in a single .cpp file, but also add a file which contains input which can be copy-pasted into the console to demonstrate creating astronomical objects and searching for astronomical objects.