OOP Constructors – Lab

Please submit your solutions (source code) to all below-described problems in Judge.

Write C++ code for solving the tasks on the following pages.
Code should compile under the C++03 or the C++11 standard.
Any code files that are part of the task are provided under the folder Skeleton.
Please follow the exact instructions on uploading the solutions for each task.

1. Echo

You are given code for a program that manages email contacts, that has a console command-based UI. The program prints hints to the user and the user enter commands to be executed.

The program has options to disable printing the hints (though commands explicitly printing information about contact will still print a result even if hints are disabled). That part of the code is missing. Your task is to implement the necessary functions and variables used by the existing code so that the program compiles and accomplishes the task specified.

You should submit a single **`.zip`** file for this task, containing ONLY the files you created.

The Judge system has a copy of the other files and will compile them, along with your file, in the same directory.

Restrictions & Hints

You can assume the code works correctly and the input will always lead to a valid sequence of commands. Focus on the hint-printing part, you can mostly ignore the rest (*though it would be a good idea to study the code to see how such programs may be implemented*).

In addition to the skeleton, you are given a Windows executable that behaves correctly so you can study the wanted behavior before implementing the necessary code.

Examples

Input	Output
y 1 ben bendover@example.com 4 e	=== Contact Manager (C) Deluxe Edition === Hints on? (y/n): --- Menu --- 1. Add contact 2. Remove contact 3. Print contact 4. Toggle hints off/on e. Exit Please enter a choice (1, 2, 3, or Q): Enter contact name and email: --- Menu --- 1. Add contact 2. Remove contact 3. Print contact 4. Toggle hints off/on e. Exit

	Please enter a choice (1, 2, 3, or Q):
n 1 ben bendover@example.com 3 ben 4 2 ben e	=== Contact Manager (C) Deluxe Edition === Hints on? (y/n): ben bendover@example.com --- Menu --- 1. Add contact 2. Remove contact 3. Print contact 4. Toggle hints off/on e. Exit Please enter a choice (1, 2, 3, or Q): Enter the name or email of a contact: ... removed ben bendover@example.com --- Menu --- 1. Add contact 2. Remove contact 3. Print contact 4. Toggle hints off/on e. Exit Please enter a choice (1, 2, 3, or Q): Exiting... Thank you for using Contact Manager (C) Deluxe!

2. Notes

You are given code for a program that reads musical note names in the solfège naming convention (Do–Re–Mi–Fa–Sol–La–Si) and translates them into **`NoteName`** objects following the English naming convention (C–D–E–F–G–A–B), however, the provided code is missing the translation logic.

Your task is to study the way the provided code uses the translation logic and implement the translation logic so that the code compiles successfully and accomplishes the task described.

You should submit a single **`.zip`** file for this task, containing ONLY the files you created.

The Judge system has a copy of the other files and will compile them, along with your file(s), in the same directory.

Restrictions & Hints

You can assume the code works correctly and the input will always lead to a valid sequence of commands. Focus on the hint-printing part, you can mostly ignore the rest (*though it would be a good idea to study the code to see how such programs may be implemented*).

In addition to the skeleton, you are given a Windows executable that behaves correctly so you can study the wanted behavior before implementing the necessary code.

Examples

Input	Output
Do Re Mi Fa Sol La Si unknown end	C D E F G A B ?

3. Vectors

You are given code for a program that reads **Vector**s from the console, sorts them in reverse order of their length (the Vector class defines methods for length calculation), and prints them (longest-first). For that, it uses a **multiset** and supplies an additional template parameter that instructs the multiset on how to compare two Vector objects.

Your task is to study the way the provided code and create types that allow the appropriate comparison of **Vector** objects, such that the resulting code accomplishes the task described.

You should submit a single **.zip** file for this task, containing ONLY the files you created.

The Judge system has a copy of the other files and will compile them, along with your file(s), in the same directory.

Examples

Input	Output
3 1 3 4 5 -5 10	-5 10 4 5 1 3