# 05. Code Organization and C++ Templates

Write C++ code for solving the tasks on the following pages.

Code should compile under the C++03 or the C++11 standard.

Submit your solutions here: <https://judge.softuni.bg/Contests/1187/05-Code-Organization-and-CPlusPlus-Templates-Homework> (select “Compete” when prompted)

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.

NOTE: the Judge system treats each .cpp file as a compilation unit, compiles each such file and links them together to create the final executable, which is checked against the tests.

# Task 5 – Sorting

NOTE: this task is the same as **Task 3 – Parser**, however the output is sorted depending on the object.

You are given code that reads objects from the console, each from a separate line, until a “**stopLine**” is entered. The format of the input is the following

* A character called **type**, which determines the type of each object – i for **integer**, w for string, and s for Song
* A line representing the **stopLine**
* Lines, each containing a single object, until a line matching the **stopLine** is reached – the program should not read anything after the **stopLine**.

The program then uses a set to store and then print all **unique** the objects on a single line, **sorted in descending order**, separated by single spaces. Descending order is different for each object type:

* For integers, it is the same as the standard C++ ordering of integers, in reverse, i.e. larger numbers are placed before smaller numbers
* For strings, it is the reverse of the C++ lexicographical order of strings
* For Songs, it is a reverse ordering by the lengthSeconds field, i.e. longer Songs are placed before shorter Songs

Your task is to study the available code and create any necessary files for the program to compile and work correctly.

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 files, in the same directory.

### Restrictions

Each type of object supports stream insertion operator<<

Each object is valid, meaning that each line representing an object can be turned into an object of the appropriate type through stream extraction operator>>

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| i  7  1  2  3  4  5  6  7  8 | 6 5 4 3 2 1 |
| s  ...  caught-somewhere-in-time 446  superpalav 246  ...  this is not printed | caught-somewhere-in-time 446 superpalav 246 |