# 09. C++ OOP Pure Virtual Members and Multiple Inheritance

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/1281/09-Cpp-OOP-Pure-Virtual-Members-and-Multiple-Inheritance> (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 3 – Files

You are given code which reads information File and Directory objects in a file system, each of which has an id, and each of which has a parent – the object which contains it.

NOTE: For this task, only files will have parents (but similar logic will be used in other tasks in this homework assignment).

Operations with the files and directories are done with reference to their **id**, and there are several types of operations:

* file – create a File object with a filename and **contents** (a sequence of characters, stored in a string)
* directory– create a Directory object with a name
* copy– move a file into a directory, only 1 such operation will be done for any file, and it will always contain a file id to move and a directory id to move to
* size – prints the **size in bytes** of a File or Directory. The size of a File is equal to the size (number of characters) of its **contents**. The size of Directory is equal to the **sum of the sizes** of the File objects in it.
* path– prints the path of a File or Directory. The path is the sequence of parents for the File/Directory, separated by "/", followed by the name of the File/Directory. For this task, only Files will have parents, meaning that paths for a directory will always be just its name.
* print – prints the contents of a File – will only be called with ids of already existing Files

The provided code is missing the definitions for the File and Directory classes – you should implement them. Also, you should study the code and see what inheritance hierarchy is used to represent the file system and implement any other necessary classes/interfaces and functions.

You should submit a single .zip file for this task, containing ONLY the file(s) 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

The input will always contain correct operations – i.e. any object used by an operation will have already been created by the file or directory operations. There will be no invalid ids, no copy or print operations on Directory objects (but the destination of a copy operation will always be a Directory).

The provided code handles input/output and operation management – you should focus on implementing the classes it uses.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| file 1 example.txt some example bytes as text  file 2 otherFile.txt other text  path 1  directory 3 examples  path 3  copy 1 3  path 1  print 1  path 2  end | example.txt  examples  examples/example.txt  some example bytes as text  otherFile.txt |
| file 1 example.txt some example bytes as text  file 2 otherFile.txt other text  size 1  size 2  directory 3 examples  size 3  copy 1 3  size 3  copy 2 3  size 3  end | 26  10  0  26  36 |