# 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 4 – Tree

Like in **Task 3** and **Task 4** you are given code which reads information File and Directory objects in a file system, however it uses an Explorer class to create, cut & paste, create shortcuts and navigate between them.

You are tasked with implementing the Explorer class so that it supports the operations below. After all operations by the explorer are done, the tree view logic from Task 4 is used to print the resulting file system.

The Explorer supports the following operations:

* mf – create a File object with a filename and **contents** (a sequence of characters, stored in a string), in the current directory
* md – create a Directory object with a name, in the current directory
* cut– prepare and object from the current directory to be moved. Can be called multiple times and each time it adds an object to be moved to a “clipboard”
* paste– moves the objects from the clipboard to the current directory. The object is removed from its current parent and placed in the current directory (shortcuts to the object remain unchanged)
* sc– creates a “shortcut” to a file or directory. Shortcuts do not move the object (i.e. the object remains in the directory it was originally, but it also appears in the shortcuts). Shortcuts are listed the same way as in Task 4, and no other operations will access shortcuts (i.e. no navigation to them, no copying, etc.)
* cd – changes the current directory. Receives a single path parameter, which indicates the **name of a directory**, inside the **current directory**, to which to navigate, or the string "..", which indicates **the parent of the current directory** (NOTE: this is like the DOS cd command, however you do not need to implement complex path parsing)

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. There will be no invalid or duplicate names, no invalid cd/cut/sc operations.

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

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| mf example.txt some example bytes as text  md examples  cut example.txt  cd examples  paste  md nested  cd nested  mf otherFile.txt other text  cd ..  cd ..  md rootDir  mf rootFile.txt this file is in the file system root  end | examples  --->example.txt  --->nested  --->--->otherFile.txt  rootDir  rootFile.txt |
| mf example.txt some example bytes as text  md examples  cut example.txt  cd examples  paste  md nested  sc nested  cd nested  mf otherFile.txt other text  sc otherFile.txt  mf rootFile.txt this file is in the file system root  cut rootFile.txt  cd ..  cd ..  md rootDir  paste  sc rootDir  mf noDot can't use name to check if directory or file :)  cut noDot  cd examples  cd nested  paste  end | [shortcuts]  --->nested  --->--->noDot  --->--->otherFile.txt  --->otherFile.txt  --->rootDir  examples  --->example.txt  --->nested  --->--->noDot  --->--->otherFile.txt  rootDir  rootFile.txt |