Network register I.

Submission deadline:	**2019-04-28 23:59:59**
Late submission with malus:	**2019-06-30 23:59:59** (Late submission malus: 100.0000 %)
Evaluation:	**4.0000**
Max. assessment:	**4.0000** (Without bonus points)
Submissions:	4 / 20 Free retries + 20 Penalized retries (-2 % penalty each retry)
Advices:	2 / 2 Advices for free + 2 Advices with a penalty (-10 % penalty each advice)

The problem is to design and implement classes that simulate a database of networked computers. We need to store information about networks (`CNetwork`), computers (`CComputer`) and their components: `CCPU`, `CMemory`, and `CDisk`.

This assignment is focused on class design, where inheritance, polymorphism and abstract methods are used. If these OOP paradigms are used correctly, the implementation is short and clean. On the other hand, if the design is wrong, the implementation will be lengthy with repeated code. Try to identify base class and subclasses, use inheritance.

The classes and the interface:

`CNetwork`
: represents a network. The interface is:

- constructor with the network name parameter,
- destructor, copy constructor and operator = (if the automatically generated are not correct),
- method `AddComputer` which adds another computer to the list,
- method `FindComputer` which returns a pointer to `CComputer` object with the given name, or an empty pointer if the computer of that name does not exist,
- output operator which displays the network in the format shown below. The computers are listed in the order they were added into the network object.

`CComputer`
: represents a computer. The interface is:

- constructor with computer name parameter (string),
- destructor, copy constructor and operator = (if the automatically generated are not correct),
- method `AddComponent` which adds another component to the list,
- method `AddAddress` which adds another address to the list of addresses,
- output operator which displays the computer in the format shown below. The addresses are listed first (in the order they were added), followed by the list of components (again in the order they were added).

`CCPU`
: represents a CPU. The interface is:

- constructor with the number of cores (int) and frequency (int, MHz) parameters,
- destructor, copy constructor and operator = (if the automatically generated are not correct).

`CMemory`
: represents a RAM memory. The interface is:

- constructor with the memory size (int, in MiB),
- destructor, copy constructor and operator = (if the automatically generated are not correct).

`CDisk`
: represents a storage. The interface is:

- constructor with the storage type (symbolic constant `SSD` or `MAGNETIC`) and disk size (int, in GiB),
- destructor, copy constructor and operator = (if the automatically generated are not correct),
- method `AddPartition` which adds another partition to the disk description. The method will take two parameters: partition size (int, in GiB) and a partition description (string). The partitions are listed in the order they were added.

Submit a source code with the implementation of classes `CNetwork`, `CComputer`, `CCPU`, `CMemory`, and `CDisk`. All required auxiliary declarations/functions shall be included in the source file submitted. The #include preprocessor definitions and your tests shall be placed in the conditional compile blocks (as in the attached sample).

Notes

- Use the typecast operators (`dynamic_cast`) with caution. The reference implementation does not use any of the RTTI based operator (i.e. it does not use `dynamic_cast`, nor it uses `typeid`). In general, if RTTI based operators are used too often, your design is probably sloppy. A code with RTTI based operators tends to have many branches, is difficult to understand, and is difficult to extend. Pay a special care to the design and use polymorphism instead of RTTI.
- Please note there is no `typeinfo` header file included. Thus, your implementation cannot use `typeid`.
- Your implementation must use classes that form an inheritance hierarchy. This problem is suited for a solution that makes use of both inheritance and polymorphism. Moreover, the testing will reject solutions which do not use inheritance, polymorphism, and dynamic binding (a compile-time error will be reported).
- The output is in the form of a tree (even that the tree may be only 3-level here). Please note that the vertical lines are suppressed where not needed. Moreover, the last junction is represented by a single backslash character.

A correct solution of this homework may be used for code review. A solution is considered correct if it passes all mandatory tests for 100%.

Sample data: **Download**

☐ **Reference**

4	**2019-04-28 14:44:28**	**Download**
Submission status:	Evaluated	
Evaluation:	4.0000	

- **Evaluator: computer**
  - Program compiled
  - Test 'Basic test with sample commands': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 0.000 s (limit: 3.000 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Class design test': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 0.000 s (limit: 3.000 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Random test': success
    - result: 100.00 %, required: 50.00 %
    - Total run time: 0.026 s (limit: 3.000 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Copy constructor test': success
    - result: 100.00 %, required: 50.00 %
    - Total run time: 0.001 s (limit: 2.974 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'operator= test': success
    - result: 100.00 %, required: 50.00 %
    - Total run time: 0.002 s (limit: 2.973 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Random test + mem test': success
    - result: 100.00 %, required: 50.00 %
    - Total run time: 0.047 s (limit: 4.000 s)
    - Mandatory test success, evaluation: 100.00 %
  - Overall ratio: 100.00 % (= 1.00 * 1.00 * 1.00 * 1.00 * 1.00 * 1.00)
- Advices used: 2
- Penalty due to advices: None (2 <= 2 limit)
- Total percent: 100.00 %
- Total points: 1.00 * 4.00 = 4.00

SW metrics:

	Total	Average	Maximum	Function name
Functions:	**34**	--	--	--
Lines of code:	**321**	**9.44 ± 22.29**	129	`main`
Cyclomatic complexity:	**61**	**1.79 ± 2.73**	17	`operator`

3	**2019-04-28 14:39:04**	**Download**
Submission status:	Evaluated	
Evaluation:	1.9615	

- **Evaluator: computer**
  - Program compiled
  - Test 'Basic test with sample commands': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 0.000 s (limit: 3.000 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Class design test': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 0.000 s (limit: 3.000 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Random test': success
    - result: 100.00 %, required: 50.00 %
    - Total run time: 0.026 s (limit: 3.000 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Copy constructor test': success
    - result: 100.00 %, required: 50.00 %
    - Total run time: 0.001 s (limit: 2.974 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'operator= test': success
    - result: 50.00 %, required: 50.00 %
    - Total run time: 0.002 s (limit: 2.973 s)
    - Mandatory test success, evaluation: 50.00 %
    - ☐ Failed (invalid output)
  - Test 'Random test + mem test': success
    - result: 98.08 %, required: 50.00 %
    - Total run time: 0.060 s (limit: 4.000 s)
    - Mandatory test success, evaluation: 98.08 %
    - Failed (invalid output) **[Unlock advice (19.47 KiB)]**
  - Overall ratio: 49.04 % (= 1.00 * 1.00 * 1.00 * 1.00 * 0.50 * 0.98)
- Advices used: 1
- Penalty due to advices: None (1 <= 2 limit)
- Total percent: 49.04 %
- Total points: 0.49 * 4.00 = 1.96

SW metrics:

	Total	Average	Maximum	Function name
Functions:	**34**	--	--	--
Lines of code:	**320**	**9.41 ± 22.30**	**129**	`main`
Cyclomatic complexity:	**61**	**1.79 ± 2.73**	**17**	`operator`

2	**2019-04-27 22:33:05**	**Download**
Submission status:	Evaluated	
Evaluation:	0.0000	

- **Evaluator: computer**
  - Program compiled
  - Test 'Basic test with sample commands': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 0.000 s (limit: 3.000 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Class design test': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 0.000 s (limit: 3.000 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Random test': failed
    - result: 41.88 %, required: 50.00 %
    - Total run time: 0.027 s (limit: 3.000 s)

- Mandatory test failed, evaluation: 0.00 %
- Failed (invalid output) **[Unlock advice (1.54 KiB)]**
- Failed (invalid output) **[Unlock advice (3.85 KiB)]**
- Failed (invalid output) **[Unlock advice (768 B)]**
- Failed (invalid output) **[Unlock advice (2.79 KiB)]**
- Failed (invalid output) **[Unlock advice (1.90 KiB)]**
- Failed (invalid output) **[Unlock advice (3.11 KiB)]**
- Failed (invalid output) **[Unlock advice (1.36 KiB)]**
- Failed (invalid output) **[Unlock advice (618 B)]**

- Overall ratio: 0.00 % (= 1.00 * 1.00 * 0.00)

- Advices used: 1
- Penalty due to advices: None (1 <= 2 limit)
- Total percent: 0.00 %
- Total points: 0.00 * 4.00 = 0.00

SW metrics:

	Total	Average	Maximum	Function name
Functions:	**34**	--	--	--
Lines of code:	**290**	**8.53 ± 20.87**	**126**	`main`
Cyclomatic complexity:	**56**	**1.65 ± 2.06**	**13**	`operator`

1	**2019-04-27 17:06:27**	**Download**
Submission status:	Evaluated	
Evaluation:	0.0000	

- **Evaluator: computer**
  - Program compiled
  - Test 'Basic test with sample commands': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 0.000 s (limit: 3.000 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Class design test': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 0.000 s (limit: 3.000 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Random test': failed
    - result: 40.63 %, required: 50.00 %
    - Total run time: 0.027 s (limit: 3.000 s)
    - Mandatory test failed, evaluation: 0.00 %
    - Failed (invalid output) **[Unlock advice (1.54 KiB)]**
    - Failed (invalid output) **[Unlock advice (1.33 KiB)]**
    - ☐ Failed (invalid output)
    - Failed (invalid output) **[Unlock advice (960 B)]**
    - Failed (invalid output) **[Unlock advice (2.44 KiB)]**
    - Failed (invalid output) **[Unlock advice (1.09 KiB)]**
    - Failed (invalid output) **[Unlock advice (2.16 KiB)]**
    - Failed (invalid output) **[Unlock advice (924 B)]**
  - Overall ratio: 0.00 % (= 1.00 * 1.00 * 0.00)
- Total percent: 0.00 %
- Total points: 0.00 * 4.00 = 0.00

SW metrics:

	Total	Average	Maximum	Function name
Functions:	**34**	--	--	--
Lines of code:	**289**	**8.50 ± 20.88**	**126**	`main`
Cyclomatic complexity:	**56**	**1.65 ± 2.06**	**13**	`operator`