Creative Software Programming, Assignment 9-1

Handed out: Nov 10, 2021

Due: 23:59, Nov 10, 2021 **(NO SCORE for late submissions!)**

- *Only files submitted by **git push to this course project at** https://hconnect.hanyang.ac.kr (<Year>_<Course no.>_<Class code>/<Year>_<Course no.>_<Student ID>.git) will be scored.*
- *Place your files under the directory structure **<Assignment name>/<Problem no.>/<your files>** just like the following example.*
  - ***DO NOT push CMake intermediate output files** (CMakeCache.txt, cmake_install.cmake, Makefile, CMakeFiles/*).*
  - ***DO NOT use cmake_minimum_required()** command in CMakeLists.txt.*

```
+ 2020_ITE0000_2019000001
  + 2-1/
    + 1/
      - 1.cpp
      - CMakeLists.txt
    + 2/
      - 2.cpp
      - CMakeLists.txt
    + …
```

- *The submission time is determined not when the commit is made **but when the git push is made**.*
- *Your files must be committed to the **master branch**. Otherwise, it will not be scored.*
- *Your program should output correct results even for inputs other than those used in the example.*
- *Basically, assignments are scored based on the output results. If it is not possible to check whether a requirement is implemented because the output is not correct, no score is given for the requirement, even if it is implemented internally. However, even if the output result is correct, no score is given for a requirement if the internal implementation does not satisfy the requirement.*

1. Write a program that works as follows:
    A. Class C inherits from class B, class B inherits from class A.
    B. Each class has a public member function test().
        i. A::test() returns a string "A::test()".
        ii. B::test() returns a string "B::test()".
        iii. C::test() returns a string "C::test()".
    C. Create objects of class A, B, and C by new operator and put them into std::vector<A*> arr in the order of user input.
    D. 'quit': Call the test() function of each element of arr to show the execution result as shown below. Each element of arr **must be deallocated** after use.
    E. Do not use the type casting operator throughout the code.
    F. Input: Class names as creation commands, exit by 'quit'.
    G. Output: The result for calling test() functions
    H. Files to submit:
        i. A C++ source file
        ii. A CMakeLists.txt to generate the executable

```
$ ./classes
A
B
C
B
C
quit
A::test()
B::test()
C::test()
B::test()
C::test()
$
```

2. Write a program that works as follows:
    A. Class C inherits from class B, class B inherits from class A.
    B. Each class has a public member function getTypeInfo().

i. A::getTypeInfo() returns a string of “This is an instance of class A”.

ii. B::getTypeInfo() returns a string of “This is an instance of class B".

iii. C::getTypeInfo() returns a string of “This is an instance of class C".

C. Define the following two functions in the global scope. Both functions print out strings obtained by calling getTypeInfo() of the object passed as argument.

i. void printObjectTypeInfo1(A* object)

ii. void printObjectTypeInfo2(A& object)

D. Create objects of class A, B, and C by new operator and put them into std::vector<A*> arr in the order of user input.

E. ‘quit’: Call printObjectTypeInfo1() and printObjectTypeInfo2() at the end of the code by passing each element of arr as an argument. Each element of arr **must be deallocated** after use.

F. Do not use the type casting operator throughout the code.

G. Input: Class names as creation commands, exit by ‘quit’.

H. Output: The result for printObjectTypeInfo1() and printObjectTypeInfo2()

I. Files to submit:

i. A C++ source file

ii. A CMakeLists.txt to generate the executable

```
$ ./print_info
A
C
B
quit
This is an instance of class A
This is an instance of class A
This is an instance of class C
This is an instance of class C
This is an instance of class B
This is an instance of class B
$
```

3. Write a program that works as follows:

   A. Class C inherits from class B, class B inherits from class A.

   B. Add member variables.

      i. Add memberA, a private member variable of type int* to class A.

      ii. Add memberB, a private member variable of type double* to class B.

      iii. Add memberC, a private member variable of type std :: string* to class C.

   C. Constructors

      i. The constructor of class A takes an [int] argument, allocates memberA with new, stores the int argument value in the allocated space, and prints the string "new memberA".

      ii. The constructor of class B takes a [double] argument, allocates memberB using new, stores the double argument value in the allocated space, and prints out the string "new memberB". And it calls the constructor of class A from the initialization list, passing an integer 1 as the argument, to initialize memberA.

      iii. The constructor of class C takes a [const std::string&] type argument, allocates memberC with new, stores the string in the allocated space, and prints the string “new memberC”. And it calls the constructor of class B from the initialization list, passing a double value 1.1 as the argument, to initialize memberB.

   D. Destructors

      i. The destructor of class A uses delete to free memberA and prints "delete memberA".

      ii. The destructor for class B uses delete to free memberB and prints "delete memberB".

      iii. The destructor of class C uses delete to free memberC and prints "delete memberC".

   E. A, B, and C all have member functions void print().

      i. A::print() prints out the data stored in the space pointed to by memberA.

      ii. B::print() calls A::print() first, and then prints out the data stored in the space pointed to by memberB.

      iii. C::print() calls B::print() first, and then prints out the data stored in the space

pointed to by memberC.

F. Take an integer (for class A), real number (for class B), or string (for class C) with its corresponding class name from the user, and then create an object of corresponding class by new operator with user inputs and put it into std::vector<A*> arr.

G. 'quit': Call the print() function of each element of arr. Each element of arr **must be deallocated** after use.

H. Do not use the type casting operator throughout the code.

I. Input: "A" and a integer, or "B" and a real number, or "C" and a string, exit by 'quit'.

J. Output: The result for print(), and creating and destructing objects.

K. Files to submit:

   i. A C++ source file

   ii. A CMakeLists.txt to generate the executable

```
$ ./print_member
A 20
new memberA
B 3.14
new memberA
new memberB
C test
new memberA
new memberB
new memberC
quit
*memberA 20
*memberA 1
*memberB 3.14
*memberA 1
*memberB 1.1
*memberC test
delete memberA
delete memberB
delete memberA
delete memberC
delete memberB
delete memberA
$
```