COMP 2012 Midterm Exam - Spring 2018 - HKUST

Date: March 24, 2018 (Saturday)
Time Allowed: 2 hours, 1–3 pm
Instructions:

1. This is a closed-book, closed-notes examination.
2. There are **7** questions on **25** pages (including this cover page and 3 blank pages at the end).
3. Write your answers in the space provided in black/blue ink. *NO pencil please, otherwise you are not allowed to appeal for any grading disagreements.*
4. All programming codes in your answers must be written in the ANSI C++ version as taught in the class.
5. For programming questions, unless otherwise stated, you are **NOT** allowed to define additional structures, classes, helper functions and use global variables, **auto**, nor any library functions not mentioned in the questions.

Student Name	
Student ID	
Email Address	
Venue & Seat Number	

For T.A. Use Only

Problem	Score
1	/ 10
2	/ 8
3	/ 10
4	/ 8
5	/ 9
6	/ 25
7	/ 30
Total	/ 100

Problem 1 [10 points] True or false

Indicate whether the following statements are *true* or *false* by circling **T** or **F**. You get 1.0 point for each correct answer, −0.5 for each wrong answer, and 0.0 if you do not answer.

T **F** (a) There is compilation error in the following code segment.

```
class A {
  public:
    int& getA() const {
      return a;
    }
    int calc(int b) const {
      return getA() + b;
    }
  private:
    const int a{10};
};
```

T **F** (b) For a const data member of a class, if it is already initialized by a default member initializer in the class definition, it cannot be initialized again by any constructor of the class.

T **F** (c) Default argument for a given parameter of a function should NOT be specified in both .h and .cpp files.

T **F** (d) The compiler always generates a default constructor for a class if no such constructor has been implemented.

T **F** (e) The initialization order of non-static data members can be controlled by the order of their appearance in the member initialization list.

T **F** (f) If an object of a user-defined type is returned by a function by value, the returned object cannot be further modified. In contrast, if the object is returned by reference, it can be further modified.

T **F** (g) The output of the following program is `~C~A~B`

```
#include <iostream>
using namespace std;

class A {
  public:
    ~A() { cout << "~A"; }
};
class B {
  public:
    ~B() { cout << "~B"; }
};
class C : public B {
  public:
    ~C() { cout << "~C"; }
    A a;
};

int main() {
  C* c = new C;
  delete c;
}
```

T **F** (h) There is NO compilation error in the following program.

```
class Base {
  private: int a;
  public: Base(int a) { }
};

class Derived : public Base {
  private: int b;
  public: Derived(int a, int b) : b(b), Base(a) {}
};

int main() {
  Derived obj(1, 2);
}
```

T **F** (i) With public inheritance, one may always assign a pointer of a base class `B` to an object of its derived class `D`:

```
D d;
B* bp = &d;
```

regardless of whether `B` is a direct base class, an indirect base class, or an abstract base class of `D`.

T **F** (j) C++ does not allow an abstract base class to be derived from another abstract base class.

Problem 2 [8 points] Classes and Objects

```
1  #include <iostream>
2  #include <cstring>
3  using namespace std;
4
5  class Word
6  {
7    private:
8      int freq = 0; char* str = nullptr;  // You are NOT allowed to modify this line
9
10   public:
11     Word() = default;         // Default constructor
12
13     Word(char* s, int f = 1) // Initialize with the given C string s and frequency f
14     {
15         freq = f;
16         str = s;
17         cout << "conversion: "; print();
18     }
19
20     ~Word() // You are NOT allowed to modify the destructor function
21     {
22         cout << "destructor: "; print();
23         delete [] str;
24     }
25
26     void print() const // Print out the stored C string and frequency
27     {
28         cout << str << " ; " << freq << endl;
29     }
30 };
31
32 int main() // You are NOT allowed to modify the main function
33 {
34     char s[] = "stress";
35     Word x, y, z {s};
36     x = z;
37     return 0;
38 }
```

The program above will compile but it runs with 3 run-time errors during the destruction of the 3 `Word` objects, `z`, `y`, and `x` in that order. *Some remarks:*

- *You are NOT allowed to change the meaning of any given member function.*
- *You are NOT allowed to modify the main function nor the destructor function of the Word class in your answers to this question.*
- *You don't get ANY marks if you only give the line numbers without correction for parts (a) and (b), or explanation for part (c).*

(a) [3 points] Identify the statement, by giving its line number in the program, that causes the run-time error when `z` is destructed. Re-write the concerned statement to eliminate this error.

Answer:

(b) [3 points] However, even after you fix the error in part (a), it still will run into another run-time error when `y` is destructed. Again, identify the statement, by giving its line number, that causes the error, and then re-write the concerned statement to eliminate this 2nd error.

Answer:

(c) [2 points] To your dismay, even after you fix the 2 errors in part (a) and (b), your program still will run into the 3rd run-time error when `x` is destructed. Identify, for the last time, the statement, by giving its line number, that causes this last error. This time, you only need to explain how the error is produced and you don't need to fix it.

Answer:

Problem 3 [10 points] Const-ness

In the following program, for the 10 statements ending with the following comment:

```
/* Error:   Yes   /   No   /   Don't know   */
```

decide whether the statement is syntactically INCORRECT - that is, it will produce compilation error(s). Circle "Yes" if it will give compilation error and "No" otherwise.
You get 1 point for each correct answer, -0.5 for each wrong answer, and 0.0 if you do not answer by circling "Don't know".

```
#include <iostream>
using namespace std;

int main() {
  const int** ptr1 = new int*;         /* Error:  Yes  /  No  /  Don't know  */

  int** const ptr2 = new int*;         /* Error:  Yes  /  No  /  Don't know  */

  const int** ptr3 = new const int*;   /* Error:  Yes  /  No  /  Don't know  */

  const int** ptr4 = new int const*;   /* Error:  Yes  /  No  /  Don't know  */

  const int** ptr5 = new int* const;   /* Error:  Yes  /  No  /  Don't know  */

  // To evaluate the correctness of the following statements, you may
  // assume that the incorrect statements (if any) are fixed so that all
  // pointer variables declared above are correctly allocated.

  *ptr1 = new int;
  **ptr1 = 1;                          /* Error:  Yes  /  No  /  Don't know  */
  *ptr2 = new int;
  **ptr2 = 2;                          /* Error:  Yes  /  No  /  Don't know  */
  *ptr3 = new int;
  **ptr3 = 3;                          /* Error:  Yes  /  No  /  Don't know  */
  *ptr4 = new int;
  **ptr4 = 4;                          /* Error:  Yes  /  No  /  Don't know  */
  *ptr5 = new int;
  **ptr5 = 5;                          /* Error:  Yes  /  No  /  Don't know  */

  // Assume memory de-allocations are purposely not done here.
  return 0;
}
```

Problem 4 [8 points] Order of Constructions and Destructions

Write down the output of the following program when it is run.

```
#include <iostream>
using namespace std;

class Furniture
{
  public:
    Furniture() { cout << "F "; }
    ~Furniture() { cout << "~F "; }
};

class Seat : public Furniture
{
  public:
    Seat() { cout << "S0 "; }
    Seat(int n) { cout << "Sn "; }
    ~Seat() { cout << "~S "; }
};

class Display
{
  public:
    Display() { cout << "D "; }
    ~Display() { cout << "~D "; }
};

class Machine
{
  public:
    Machine() { cout << "M "; }
    ~Machine() { cout << "~M "; }
};

class Vehicle : public Machine
{
    Display display;
  public:
    Vehicle() { cout << "V "; }
    ~Vehicle() { cout << "~V "; }
};

class Car : public Vehicle
{
    Seat seat;
  public:
    Car() { cout << "C "; seat = Seat(2) ; }
    ~Car() { cout << "~C "; }
};

int main() { Car x; cout << endl; return 0; }
```

Answer:

Problem 5 [9 points] Inheritance

The following program contains 6 ERRORS (syntax errors, logical errors, etc.). Study the program carefully, identify each error by writing down the line number where the error occurs, and **explain** why it is an error. You don't need to write codes to fix the errors.

Remark: You don't get ANY marks if you only give the line numbers without explanation.

```
1  #include <iostream>
2  using namespace std;
3
4  class Base {
5    private:
6      int* p { nullptr };
7    public:
8      Base() = default;
9      Base(int n = 0) { // **** Assume line #9 has NO error. ****
10       p = (n) ? new int[n] : new int;
11     }
12     ~Base() {
13       delete [] p;    // **** Assume line #13 has NO error. ****
14     }
15 };
16
17 class Derived : public Base {
18   private:
19     int d(1);
20     int* p = nullptr;
21   public:
22     Derived() : p(new int) {}
23     Derived(int n, int d) : Base(n), d(d), p(new int) {}
24     Derived(const Derived& d) = delete;  // **** Assume line #24 has NO error. ****
25     Derived& func(Derived& d) {
26       d.d = 10;
27       return *this;
28     }
29     ~Derived() { delete p; }
30 };
31
32 int main() {
33   Base b { 2 };
34   Derived d1(1, 2);
35   Derived d2(d1);
36   Derived d3(2, 4);
37   Derived d4 = d3.func(d2);
38   Base* bPtr = new Derived;
39   delete bPtr;
40   return 0;
41 }
```

Error#	Line#	Explanation
1		
2		
3		
4		
5		
6		

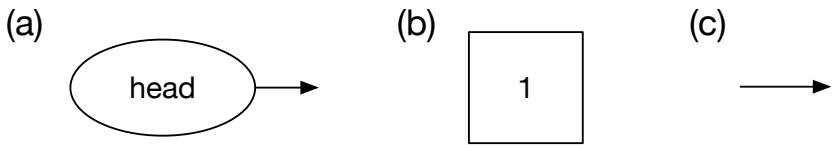
Problem 6 [25 points] Classes and Objects

Based on the given information in the question, complete the following parts.

(a) [5 points] Write down the outputs of the testing program up to and including Line 13.

Answer:

(b) [4 points] Using the symbol representation below, draw the resultant structure of the list created by the testing program.



- Use (a) to indicate the pointer called head, which points to the first node of the list.
- Use (b) to represent the node with data value 1.
- Use (c) to show the pointer pointing from one node to another node.

Answer:

(c) [16 points] Below is a partial implementation of class '`SpecialList`' in a separate file called "`SpecialList.cpp`". Complete the implementation of its

(i) destructor and

(ii) copy constructor (** deep copy is required **).

Note that you must use recursion to perform deep copy in the copy constructor. You may add any **non-member** helper function to achieve this. If you do not use recursion, you can at most get half of the full marks.

```
/* File: SpecialList.cpp */

#include "SpecialList.h"

/*
 * TODO: (i) Complete the implementation of the destructor. [6 points]
 *
 */
```

```
/* File: SpecialList.cpp */

/*
 * TODO: (ii) Complete the implementation of the copy constructor. You must use
 * recursion to perform deep copy, and you can develop any helper function to
 * achieve this. [10 points]
 *
 * If you do not use recursion, you can at most get half of the full marks.
 *
 */
```

Problem 7 [30 points] Inheritance, Polymorphism and Dynamic Binding

Based on the given information in the question, complete the following parts.

(a) [16 points] Implement the following missing member functions of the class '`Document`' in a separate file called "`Document.cpp`".

- `Document(const string* authors, unsigned int numAuthors)`
- `Document(const Document& d)`
- `~Document()`
- `void addAuthor(const string& name)`
- `void print() const`

Answer: `/* File "Document.cpp" */`

/*** Continue Your Answer For Question 7(a) On This Page ***/

(b) [5 points] Implement the following missing member functions of the class 'Book' in a separate file called "Book.cpp".

- `Book(const string& title, unsigned int edition,`
  `const string authors[], unsigned int numAuthors)`
- `void print() const`

Answer: `/* File "Book.cpp" */`

(c) [9 points] Implement the following missing member functions of the class '`Email`' in a separate file called "`Email.cpp`".

- `Email(const string& subject, const string* authors, unsigned int numAuthors, const string& toList)`
- `Email(const Email& e)`
- `void addTo(const string& name)`
- `void print() const`

Answer: `/* File "Email.cpp" */`

---------- END OF PAPER ----------