警 示 《中山大学授予学士学位工作细则》第六条

考 试 作 弊 不 授 予 学 士 学 位

计算机科学系　　2012 下学期

《程序设计 II》期末考试试题(A)

任课教师：　吴维刚，刘晓铭，刘聪　　考试形式：闭卷　　考试时间：2 小时

年级：**12**　专业：　计科　　　姓名：____________学号：____________成绩：____________

1. Single choice selection (20 points, 1 each)

Only one choice in each question is correct, no point will be given if more than one choice is selected.

1). Which of the following about class is <u>NOT</u> correct?
A) A class is a compound data type.
B) A class is a specification of a type of objects.
C) A class is a set of objects.
D) A class has member functions and member variables.

2). Which of the following about constructors is <u>NOT</u> correct?
A) They are called automatically when objects are created.
B) There can be multiple constructors in a class.
C) A class always has a default constructor.
D) Their names are the same name as those of their classes.

3). Which of the following statement does <u>NOT</u> call the default constructor?
A) Type object;
B) Type & ref = object;
C) Type * pointer = new Type();
D) Type * array = new Type[5];

4). Which of the following statement does <u>NOT</u> call the constructor Type(int)?
A) Type object(100);
B) Type object = 100;
C) Type * pionter = new Type[100];
D)Type * array = new Type[2] {Type(100), Type(200)};

5). Which of the following statement does <u>NOT</u> call the copy constructor?
A) Type object(object2);
B) Type object = object2;
C) void someFunction(Type object) { … }
D) Type & someFunction() { … return object; }

6). Which of the following about destructors is <u>NOT</u> true?
A) They have no return type
B) Each class can have more than one destructor

C) The last function each object calls must be the destructor
D) They are called automatically to finalize objects.

7). Which of the following is NOT a constant?
A) Literals
B) The arguments of a constant function
C) Function returned results
D) Anonymous objects

8). Which of the following about constant object is NOT true?
A) A constant object cannot be assigned a new value
B) Member variables of a constant object can be assigned new values
C) Non-constant object functions of a constant object cannot be called
D) A constant object cannot be passed to the reference of a variable

9). With *text* being "AB", which of the following changes s1 to "ABAB":
A) text.assign("AB");
B) text.append("AB");
C) text = "AB";
D) text + "AB";

10). With the class *Base* and its inherited class *Inherited*, which statement will have a compilation error?
A) Base * base = Inherited();
B) Base * base = new Inherited();
C) Base base = Inherited();
D) const Base & base = Inherited();

11). Which of the following about inheritance is NOT true?
A) Private member variables cannot be accessed by sub-classes (derived classes)
B) Inheritance creates a new class by including everything in an existing class
C) A sub-class (derived class) is also called a derived class
D) A base class is also called a super-class

12). Which of the following about constructor/destructor chaining is NOT correct?
A) The constructor of an inherited class must call one of the constructors in its base class
B) The constructor in an inherited class is called before that of its base class.
C) The destructor in an inherited class is called before that of its base class.
D) A constructor need to specify a base class's constructor if the base class has no default constructor.

13). Which of the following about template is NOT true?
A) A template class is a class with type parameters
B) A sub-class (derived class) of a template class must also be a template class
C) A type parameter is an unknown type in a template class
D) A container class is usually a template class with the type parameter being the type of the elements

14). Which of the following about virtual object/instance function is NOT true?

A) An object/instance function that is virtual will be matched dynamically
B) An object/instance function that is non-virtual will not be matched dynamically
C) Defining a virtual function with the same signature as a virtual function in the super-class is called overriding
D) Any object function that overrides a virtual function in a base class is also a virtual object function

15). Which of the following about abstract class is NOT correct?

A) A class that has a pure virtual function is an abstract class.
B) The type of an object cannot be an abstract class
C) The type of an address or a reference cannot be an abstract class
D) The inherited classes of an abstract class can also be abstract classes.

16). Which of the following about objects as local variables is NOT true?

A) Objects that were constructed first will be destructed first
B) The constructor of a local variable object is called when the program executes to the line that an object is defined
C) The destructors of the objects defined in a block are called when the block completes executing
D) The destructor of the object will not be called, if the constructor of a local variable object was not called

17). Which of the following function may not be automatically built when it is not defined?

A) The default constructor
B) The copy constructor
C) The assignment operator function
D) The destructor

18). Which of the following about exception handling is NOT correct?

A) An exception is an error at runtime
B) An exception cannot be match by more than one catch-block
C) Only an exception thrown in a try-block can be caught
D) An exception must be handled where it occurs.

19). To output data into a file, it is necessary for you to

A) make sure the file is already there before opening it.
B) open it in binary mode.
C) know the format of the data.
D) convert the data to characters (or an object of type string).

20). Which of the following about static members (static variables and functions) is true?

A) Static variables are the same as object variables
B) Static variables must be defined “again” outside the class
C) All static variables can be called in functions outside the class
D) Object functions can be called within a static function without an object

2. Output analysis (20 points, 5 each)

Write the printout of the following programs.

1). Constructor and destructor

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

class People
{
public:
    People() {
        cout << "People()" << endl;
    }

    ~People() {
        cout << "~People()" << endl;
    }

    People(const People & people) {
        cout << "People(People &)" << endl;
    }
};

int main() {
    People me;
    People you(me);
}
```

2). Inheritance and virtual function

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

class Computer
{
public:
    virtual void printSpeed() const = 0;
};

class Notebook : public Computer
{
public:
    void printSpeed() const {
        cout << "Fast" << endl;
    }

    void printWeight() const {
        cout << "Heavy" << endl;
    }
};

class Tablet : public Notebook
{
public:
    void printSpeed() const {
        cout << "Slow" << endl;
    }

    void printWeight() const {
        cout << "Light" << endl;
    }
};

int main() {
    const Notebook & notebook = Tablet();
    notebook.printSpeed();
    notebook.printWeight();
}
```

3). String and vector

```cpp
#include <iostream>
#include <vector>
#include <string>
using namespace std;

void print(vector<string> & vec) {
    for (int i = 0; i < vec.size(); ++ i) {
        cout << vec[i] << endl;
    }
}

int main() {
    string text("apple");
    vector<string> vec;
    vec.push_back(text);
    text += " juice";
    vec.push_back(text);
    text = "orange";
    vec.push_back(text);
    print(vec);
}
```

4). Template and exception

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

class EmptyStackException {};

template <typename E>
class Stack
{
private:
    vector<E> impl;

public:
    void push(const E & e) {
        impl.push_back(e);
    }

    E pop() {
        if (impl.size() == 0)
            throw EmptyStackException();
        E e = impl[impl.size() - 1];
        impl.pop_back();
        return e;
    }
};

int main() {
    Stack<double> stack;
    for (int i = 0; i < 3; ++ i) {
        stack.push(0.5 + i);
    }
    try {
        while (true) {
            cout << stack.pop() << endl;
        }
    }
    catch (EmptyStackException & ex) {
        cout << "caught: EmptyStackException" << endl;
    }
}
```

3. Error correction (20 points, 5 each)

There are 5 errors in total in each of the following program.
You need to find out 10 of them, which you are most confident.
If you list more than 10 errors, only the first 10 errors will be graded.

1). Class and object

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

 class A
 {
     int value;
 public:
     void   A(int value) {
         this.value = value;
     }

     A(A &a) {
         value = a->value;
     }
     ~A(A a) {
     }
 };

 int main() {
     A a;
     return 0;
 }
```

2). Inheritance and polymorphism

```
class Base
{
    virtual print() const = 0;
}

class Inherited :: public Base
{
public:
        void print() const {
            cout << "Inherited" << endl;
            return 0;
        }
};

int main() {
    Inherited()-> print();
    return 0;
```

The second listing begins with the same two lines:

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

```
21)             }
```

3). Template and vector

```
1)      #include <iostream>
2)      #include <vector>
3)      using namespace std;
4)
5)      template <typename E>
6)      class MyQueue<E>
7)      {
8)      private:
9)              vector   impl;
10)
11)     public:
12)                 void enqueue(E &e){
13)                         impl.push_back(e);
14)                 }
15)                 E dequeue();
16)     };
17)
18)     E   MyQueue<E>::dequeue() {
19)                E e = impl[0];
20)                impl.pop_back();
21)
22)     }
23)
24)     int main() {
25)                MyQueue<double> q;
26)                q.enqueue(100);
27)                q.dequeue(0);
28)     }
```

4. Concept explanation with example (20 points, 5 each).

Please explain the following concepts with concrete examples. You can choose 4 out of all the 5 questions to answer.

1) The meaning of encapsulation in class

2) The relationship between class and object

3) The differences between shallow copy and deep copy

4) The similarities and differences between the container class vector and a C/C++ array

5. Programming (20 points).

Please complete the following program.

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

/*
 A class for the information of a classmate.
 This information includes the name and
 the address of the classmate
 */
class ClassmateInfo
{
public:
    string name;
    string address;
};

/*
 An exception that is thrown when you want
 to get a classmate's information with his/her
 name, but the name does not exist.
 */
class NoSuchNameException {};

/*
 An exception that is thrown when you want
 to add a classmate's information, but another
 classmate's information with the same name
 already exist.
 */
class NameExistingException {};

/*
 A container class to store all information
 of your classmates.
 */
class ClassmateInfoList
{
private:
    vector<ClassmateInfo> infoList;

```

```
public:


    // YOU NEED TO IMPLEMENT THE FOLLOWING FUNCTIONS

    // Add a classmate's information into your container
    void add(const ClassmateInfo & info);

    // Get a classmate's information that matches the name
    ClassmateInfo getByName(const string & name) const;

    // Get some classmates' information that match the address
    vector<ClassmateInfo> getByAddress(const string & address) const;

};

// ***** WRITE YOUR CODE HERE *****
// ***** WRITE YOUR CODE HERE *****
// ***** WRITE YOUR CODE HERE *****

// Read a classmate's information from the console
ClassmateInfo read() {
    ClassmateInfo info;
    cin >> info.name;
    cin >> info.address;
    return info;
}

// Print a classmate's information to the console
void print(const ClassmateInfo & info) {
    cout << "name=" << info.name << " ";
    cout << "address=" << info.address << endl;
}

void run() {
    // Read a number of 'size' classmates'
    // information and add them to your container.
    int size;
    cin >> size;
    ClassmateInfoList list;
    for (int i = 0; i < size; ++ i) {
        ClassmateInfo info = read();
        list.add(info);
    }

    // Get a classmate's information by name and print
    string name;
```

```cpp
    cin >> name;
    ClassmateInfo info = list.getByName(name);
    print(info);

    // Get some classmates' information by address and print
    string address;
    cin >> address;
    vector<ClassmateInfo> vec = list.getByAddress(address);
    for (int i = 0; i < vec.size(); ++ i) {
        print(vec[i]);
    }
}

int main() {
    try {
        run();
    }
    catch (NameExistingException & ex) {
        cout << "InsertExistingNameException" << endl;
    }
    catch (NoSuchNameException & ex) {
        cout << "NoSuchNameException" << endl;
    }
}
```