About Exams

Object-Oriented Programming with C++

Types of questions

- True-or-False
- Multiple Choice (single answer)
  - Miscellaneous
- Fill in Blank (text)
  - Write the output of the given code
- Fill in Blank (program)
  - Single line code completion
- Code Completion (subjective)
  - Class design
  - Function implementation

IDE is *NOT* allowed

Buzzwords

responsibility-driven design

Inheritance(继承)

Encapsulation (封装)

iterators

overriding

coupling

cohesion

interface

template

collection classes

mutator methods

Polymorphism(多态)

Content

- C'tor and D'tor
- assignment operator
- constant function/variable
- type casting
- new & delete
- function overloading & override
- virtual function
- abstract classes & pure virtual function
- namespace
- initializer list
- static member

Content (cont)

- reference
- default parameters
- “this” pointer
- scoping of variable & storage
- exceptions
- operator overloading
- template
- inheritance
- polymorphism
- ….

Example- Multiple Choice

- Which function below would **NOT** be automatically generated by a compiler, if the user does not provide one?
- A. Constructor
- B. Destructor
- C. operator=
- D. operator++

Example - Fill in Blank (text)

The output of the code below is : 18100

```
#include <iostream>
#include <stdexcept>
class A
{
public:
   A(int age) : m_age(age) { }

   ~A() { std::cout << m_age; }

public:
   int m_age;
};
void func(A & a)
{
      if(a.m_age < 18)
         throw std::logic_error("0");
}
int main()
{
   try {
      A a(10), b(18);
      func(b);
      func(a);
      A c(60);
      func(c);
   }
   catch (const std::exception &) {
      std::cout << 0;
   }
   std::cout << std::endl;
   return 0;
}
```

Example - Fill in Blank (text)

The output of the code below is : `Derived::fun(), x = 0`

```
// C++ program To demonstrate how default arguments
// and virtual function are used together
#include <iostream>
using namespace std;
class Base {
public:
    virtual void fun(int x = 0)
    {
        cout << "Base::fun(), x = " << x << endl;
    }
};
class Derived : public Base {
public:
    virtual void fun(int x = 10) // NOTE THIS CHANGE
    {
        cout << "Derived::fun(), x = " << x << endl;
    }
};
```

```
int main()
{
    Derived d1; // Constructor


    // Base class pointer which will
    // Edit value in memory location of
    // Derived class constructor
    Base* bp = &d1;


    bp->fun(); // Calling a derived class
member function


    return 0; // Returning 0 means the progra
              // Exectued successfully
}
```

Example - Fill in Blank (text)

It is better to copy directly

```
void Student::show(){
    cout<<m_name<<" age is"<<m_age<<", score is "<<m_score<<endl;
}
```

The results are:

xiaoming age is15，score is 90.6	1 分
lilei age is16，score is 80.5	1 分

Example-Fill in Blank (program)

CNY and USD account

```
#include <iostream>
#include <stdio.h>
using namespace std;

class AccountUSD;
class AccountCNY {
private:
    double dBalance {0};
public:
    void deposit(double fAmount){ //存款函数
        dBalance += fAmount;    2 point(s)
    }

    double balance(){
        return dBalance;
    }
```

```
bool transfer(AccountCNY& b, double fAmount){
    if (dBalance < fAmount)
        return false;
    dBalance -= fAmount;
    b.dBalance +=  fAmount;    2 point(s)
    return  true;
    }

    bool transfer( AccountUSD& b    2 point(s) , double fAmount);
    friend  class AccountUSD;    2 point(s)
};
```

Example-Fill in Blank (program)

```
class AccountUSD {
private:
    double dBalance {0};
public:
    void  deposit(double fAmount){
        dBalance += fAmount;    2 point(s)
    }

    double balance(){
        return dBalance;
    }

    bool transfer(AccountUSD& b, double fAmount){
        if (dBalance < fAmount)
            return false;
        dBalance -= fAmount;    2 point(s)
        b.dBalance += fAmount;
        return  true;
    }

    bool transfer(AccountCNY& b, double fAmount);
    friend  class AccountCNY;   2 point(s)
};
```

```
bool AccountCNY::transfer(AccountUSD& b, double fAmount){
    if (dBalance < fAmount)
        return false;
    dBalance -= fAmount;
    b.dBalance +=  fAmount/6.5;    2 point(s)
    return  true;
}


bool AccountUSD::transfer(AccountCNY& b, double fAmount){
    if (dBalance < fAmount)
        return false;
    dBalance  -= fAmount;    2 point(s)
    b.dBalance +=  fAmount*6.5;    2 point(s)
    return  true;
}
```

Example-Code Completion

This is a Complex Number Calculator which performs a lot of operations which are mentioned below, to simplify our day-to-day problems in mathematics. The implementation of this calculator is done using C++ Programming Language using concepts of operator overloading in OOPs.

Problem Statement:
Write a program to build a Complex Number Calculator using C++ which can perform the following operations:

1. Read Complex Number: It asks the user to enter two real and imaginary numbers of Complex Numbers to perform different operations on the complex number.

```
Example:  Real Part value: 10
          Img Part value: 20
          Real Part value: 5
          Img Part value: 7
```

Example-Code Completion

2. Display Complex Number: if the User has entered a complex number in the above function then the function display already exists a complex number.

```
Example: Values are:
                        10+i20
                        5+i7
```

3. Addition of Complex Number: It Add two Complex Numbers in such a way that the real part of 1st complex number is added to the real part of 2nd complex number and the imaginary part of 1st complex number is added to the imaginary part of 2nd complex number which the user gives as input Complex Numbers.

```
Example: Addition is:
                        15+i27
```

Example-Code Completion

It is better to highlight it is c++ code:

```c++
#include <iostream>
#include <cmath>

// Forward declarations
class Visitor;
class Rectangle;
class Triangle;
```

```
1 #include <iostream>
2 #include <cmath>
3
4 // Forward declarations
5 class Visitor;
6 class Rectangle;
7 class Triangle;
```

Otherwise:

```
#include <iostream>
#include <cmath>

// Forward declarations
class Visitor;
class Rectangle;
class Triangle;
```

```
#include
#include
// Forward declarations
class Visitor;
class Rectangle;
class Triangle;
```