Data Structures and Object-oriented Programming
Instructor: Prof. Liwei Chan
Midterm Date:2021/04/26

Student ID: Student Name:

Part I (35%, 5% each)

1. Which of the following statements is **incorrect**?
   (a) A class can have a virtual destructor.
   (b) An instance of an abstract class cannot be created.
   (c) If we don't override the pure virtual function in derived class, then derived class also becomes abstract class.
   (d) If you created a virtual function in the base class and it is overridden in the derived class then we can't add a virtual keyword in the derived class.

2. Which of the following statements is **true** about Linked List data structure?
   (a) The size of linked lists can not be changed any time.
   (b) There is at least one null pointer that exists in a circular linked list.
   (c) Linked list is an example of dynamic type memory allocation.
   (d) It is hard to insert and delete elements in the Linked List.
   (e) Random access is allowed in a typical implementation of a Linked List.

3. What is the output of the following code?

```
void foo(int &a, int *b, int c){
    a *= 2;
    b *= 2;
    *c *= 2;
}

int main(){
    int x = 2, y = 4, z = 5;
    foo(x, y, &z);
    cout << x << ", " << y << ", " << z << endl;
    return 0;
}
```

   (a) 2, 8, 10
   (b) 4, 8, 5
   (c) 4, 4, 10
   (d) compilation error

4. Consider the following code:

```
Base b;
Derived d;
Base *b1 = new Base;
Base &b2 = d;
```

The Base class is the ancestor type of the Derived class.
Which of the following castings **does not succeed** (not error)?

(a) Base* b3 = (Base*)(&d);
(b) b = d;
(c) Derived d1 = dynamic_cast<Derived&>(b2);
(d) Derived*d2 = dynamic_cast<Derived*>(b1);

5. (5%)Which is the correct output? Assume that the member variables in class is 0 when it has not been initialized.

```
class MyClass {
public:
    MyClass(int value): b(value), a(b * 2){
    cout << b << " " << a << " ";
  }
private:
    int b;
    int a;
};
class YourClass {
public:
    YourClass(int value): d(value), c(d * 10){
    cout << d << " " << c << endl;
}
private:
  int c;
  int d;
};
int main()
{
  MyClass obj(10);
  YourClass OBJ(10);
}
```

(a) 20 10 10 20
(b) 20 10 10 0
(c) 10 20 10 20
(d) 10 20 10 0

6. Which is used to create a stream that performs both input and output operations?

(a) ofstream
(b) ifstream
(c) iostream
(d) fstream

7. What is the output of the following code?

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

```
int main(){
    int x = -1;
    char *ptr = NULL;
    try{
        if(x < 0){
            throw x;
            cout << x << “ “;
        }
        if(ptr == NULL){
            throw “ptr is NULL”;
        }
    }
    catch(...){
        cout << “Exception occurred! ” << endl;
    }
    return 0;
}
```

(a) Exception occurred!
(b) Exception occurred! -1 Exception occurred!
(c) -1 Exception occurred!
(d) Exception occurred! -1

Part II (65%)

1. (10%)Please write down the numbers of the task(s) which will cause an error.

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

class A{
public:
   int x;
protected:
   int y;
private:
   int z;
};

class B : public A{
};

class C : protected A{
    void func1(){
        x = 10;   //task 1
        y = 10;   //task 2
        z = 10;   //task 3
    }
};

class D : private A{
    void func2(){
        x = 10;   //task 4
    }
};

class E : public C{
    void func3(){
        x = 10;   //task 5
        z = 10;   //task 6
    }
};

int main(){
B q;
C r;
D s;
q.y = 10;  //task 7
r.x = 10;  //task 8
r.y = 10;  //task 9
s.x = 10;  //task 10

return 0;
}
```

Answer:
3, 6, 7, 8, 9, 10

2. (10%)Please write down the output of the following code.

```
template <typename T>
void fun(const T &x){
    static int count = 0;
    cout << "x = " << x << " count = " << count << endl;
    count++;
    return;
}

int main(){
    fun<int>(1);
    fun<int>(1);
    fun<int>(3);
    fun<double>(1.3);
    return 0;
}
```

Output:

```
x = 1 count = 0
x = 1 count = 1
x = 3 count = 2
x = 1.3 count = 0
```

3. (10%)What is output of the following code?

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

class A{
public:
    virtual void func1(){ cout << "A";}
    void func2(){ cout << "B";}};

class B : public A{
public:
    void func1(){ cout << "C";}
    void func2(){ cout << "D";}};

int main(){
    A* x;
    x = new B;
    x->func1();
    x->func2();
    A y;
    B z;
    y = z;
    y.func1();
    y.func2();

    return 0;
}
```

```
Output:
CBAB
```

4. (10%)The following table describes the relationships between different classes. Fill in the table with **Yes** or **No** which represents the accessibility.

```
class Kelly{
public:
  int Height;
  friend class Anna;
protected:
  int Age;
private:
  int Weight;
};

class Joy: private Kelly{
public:
  friend class Eric;
};
class Panda: private Joy{
public:
  friend class Goose;
};
class Daniel: public Kelly{
public:
  friend class Steve;
};
class Emma: public Daniel{
public:
  friend class Keven;
};
class Cris{};
```

	Height	Age	Weight
Kelly	Yes	Yes	Yes
Anna	Yes	Yes	Yes
Eric	Yes	Yes	No
Goose	No	No	No
Steve	Yes	Yes	No
Keven	Yes	Yes	No
Cris	No	No	No

5. (5%)Please write down the output of the following code.

```
class A {
        int i;
public:
        A(int arg) {
        i = arg;
        cout << "A's constructor called: Value of i: " << i << endl;
}
        A(){
        cout << "A's constructor called" << endl;
        }
};
class B {
        A a;
public:
        B(int x):a(x) {
        cout << "B's constructor called" << endl;;
}
};
class C : public A{
public:
        C(int value):wahaha(value){
        cout << "C's constructor called: wahaha is: "<< wahaha << endl;
}
private:
        int wahaha;
};
int main() {
        B obj(10);
        C OBJ(10);
}
```

```
Output:
A's constructor called: Value of i: 10
B's constructor called
A's constructor called
C's constructor called: wahaha is: 10
```

6. (10%)The following statements are based on the class "Coordinate". If the statement is legal, write down the output. If the statement is illegal, just write down "error"(no need to explain).

```
class Coordinate{
public:
    int x;
    int y;
    Coordinate(int x, int y): x(x), y(y){}
    Coordinate(): x(0), y(0){}
    void display(){
        cout << "( " << x << ", " << y << " )" << endl;
    }
};
```

(a) Coordinate p1;
    Coordinate p2 = p1;
    p2.display();
(b) Coordinate p3();
    p3.display();
(c) Coordinate p4 = Coordinate();
    p4.display();
(d) Coordinate p5(1, 2);
    p5 = Coordinate();
    p5.display();
(e) Coordinate p6;
    p6.Coordinate(3, 4);
    p6.display();

(0.0),   (b) error,   (c) (0,0),   (d) (0,0),   (e) error

7. (10%)Choose one of the following operator overloading implementations that can make the code **execute smoothly** and get the **matched** output. Explain why you choose the function.

```
using namespace std;
class Student{
public:
    string name;
    int* grade;
    Student(string name, int grade){
        this->name = name;
        this->grade = new int(grade);
    }
    void display(){
        cout << this->name << "\t" << *this->grade << endl;
    }
    //========================================//
    // operator overloading function here     //
    //========================================//
};

int main(void){
    Student A("Alice",90), B("Bob", 77), C("Carol", 65);
    C = B = A;
    *A.grade = 60;
    A.display();
    B.display();
    C.display();
    return 0;
}
```

```
Output:
Alice 60
Bob   90
Carol 90
```

(a)

```
Student& operator=(Student& A){
    *this->grade = *A.grade;
    return *this;
}
```

(b)

```
Student operator=(Student& A){
    *this->grade = *A.grade;
    return *this;
}
```

(c)

```
Student operator=(Student& A){
    this->grade = A.grade;
    return *this;
}
```

(d)

```
Student& operator=(Student& A){
    this->grade = A.grade;
    return *this;
}
```

C = B = A can also be represented as (C = (B = A)). B will call operator overloading function first and pass A's reference as parameter. Assign A's grade to B.
Then C will call operator overloading function and pass the return value of (B = A) as parameter. So the return type of the function is reference.

grade is an int ptr. Assign A's grade to B's grade means let A and B point to same address. (this->grade = A.grede) We can't only change A's grade. So we should change the value that grade point to. (*this->grade = *A.grade)