1. What is the difference between a shallow copy and a deep copy?

Shallow copies everything in the object, including the object's memory location. Thus, if copied new variable is modified, the original object will get modified.

On the other hand, the deep copy does not effect the new copied variable because it does copy the original address.

1. What is the rule of 3?

Rule of 3: destructor copy, constructor copy, assignment operator

When copying a constructor to another existed constructor, we need to override the assignment operator to avoid the shallow copy. In addition, we should build a destructor to delete the address element in the object. If don’t we may cause a memory leak.

1. The copy constructor for a class X has the form X(const X& b). Why is the parameter passed by reference? Explain why you can’t define a constructor of the form X(X b)?

If we do not pass by reference, the compiler will start to create a copy of “obj”. However, the compiler is not going to copy its object to copy it, but rather to create a new object again to copy it. Then it will become a infinitely recursive.

1. What would be the implication of declaring a copy constructor private?

It is for causing a compile error if we don’t need someone use a copy constructor.

1. What is the difference between destruction and deletion of an object?

Deletion of an object only delete the object, but it does not free those elements in the memory address. Thus, we need destruction not only delete the object but also free those elements in the memory address, for example, a linked list.

1. What problems could a programmer encounter if they defined a destructor for a class but no assignment operator? Illustrate your description with an example class.

It may create a memory leak.

Class example

{

public:

// Constructor

Example(string name = “example name”, int =5);

//Copy Constructor

Example(const Example & obj);

//Overloaded Assignment

Example & operator=(const Example& obj);

//Destructor

~Example();

Private:

string name;

int num;

}

Example::Example(string n, int amount)

{

name = n;

number = amount;

}

Example::Example(const Example& obj) //copy constructor

{

this.name = obj.name;

this.number = obj.number;

}

Example& Example::operator=(const Example& obj)

{

If(this==&obj){

return \*this;

}

this->name = obj.name;

this-> number = obj.number;

return \*this;

}

//Destructor

Example:: ~Example(){

}

1. What problems could a programmer encounter if they defined a destructor for a class but no copy constructor? Illustrate your description with an example class.

It will only clean out the orginal object.

1. Which objects are destroyed when the following function exits? Which values are deleted?

void f(const Fraction& a)

{

Fraction b = a;

Fraction\* c = new Fraction(3, 4);

Fraction\* d = &a;

Fraction\* e = new Fraction(7,8);

Fraction\* f = c;

delete f;

}

C

1. What error is being committed in the assignment operator for the following class?

class String

{

public:

String(const char right[]);

String& operator=(const String& right);

private:

char\* buffer;

};

String::String(const char right[])

{

len = 0;

while (right[len] != '\0')

len++;

buffer = new char[len + 1];

for (int i = 0; i < len; i++)

buffer[i] = right[i];

buffer[len] = '\0';

}

String& String::operator=(const String& right)

{

int n = right.length();

for (int i = 0; i <= n; i++)

buffer[i] = right.buffer[i];

return \*this;

}

It should use “new” to allocate a new memory address for the buffer.

Use the following class for the following problem. The only purpose of the class is to display a message both when the constructor is invoked and when the destructor is executed.

class Trace

{

public:

Trace(string n);

~Trace();

private:

string name;

};

Trace::Trace(string n) : name(n)

{

cout << "Entering " << name << "\n";

}

Trace::~Trace()

{

cout << "Exiting " << name << "\n";

}

Trace::Trace(const Test &t)

    {

this->name = t.name ;

    }

Trace& Trace::operator=( const Test &t )

{

If(this==&t){

return \*this;

}

this->name = t.name;

return \*this

}

1. Extend the class Trace with a copy constructor and an assignment operator, printing a short message in each. Use this class to demonstrate
   1. the difference between initialization

Trace t("abc");

Trace u = t;

First create a object t with normal constructor. After that, copying object t to a object u with copy constructor.

Trace t("abc");

Trace u("xyz");

u = t;

Create two objects, then use assignment operator to assignment t to u.

* 1. the fact that all constructed objects are automatically destroyed.

True.

* 1. the fact that the copy constructor is invoked if an object is passed by value to a

function.

False.

* 1. the fact that the copy constructor is not invoked when a parameter is passed

by reference.

True.

* 1. the fact that the copy constructor is used to copy a return value to the caller.True.