The final is comprehensive. It has some T/F, most are questions to read code or write code.

* The questions for modules 1-5 are similar to the midterms.

Topics: 2D arrays, binary search, insertion sort, dynamic memory allocation, C strings, C++ string class, structs, classes and objects

* The new material is module 7, so here are the review questions on modules 5, 6, 7

1. Given this class alpha:

class alpha

{

public:

alpha(int input=1) {num = input;} **//default constructors**

alpha operator++() {++num; return num;}**//operator overload ++pre**

int operator=(int x) {num = x+1; return num;}**//=overload adds 1 to the object**

friend ostream & operator<< (ostream & o, alpha & a) **//<<overload and subtract 1**

{o << a.num - 1; return o;}

private:

int num;

};

Show what value each alpha object has after each line of code, or write error if there is an error

void main()

{

alpha A; **//A.num = 1**

alpha B = A++; **//error**

**return type of ++ operator(returns int, but that's ok, //constructor will run to return an alpha object)**

**//post fix is needed by main, there is no matching operator //overload function**

A = 10; **//A.num = 11**

cout << A; **//print: 10**

A = B; **//error because there is no B**

**//if B was actually created it would work**

}

2. Given the following classes:

class base

{

public:

base(int in = 0) {ptr = new int; \*ptr = in;}

~base() {delete ptr;}

void show() {cout << \*ptr << endl;}

virtual void F(int in) {\*ptr = in;}

protected:

int \* ptr;

};

class derived : public base

{

public:

derived(int in) {num = in;}

void show() {cout << num << endl;}

void F(int in) {num = in;}

private:

int num;

};

void main()

{

base B(5); **//\*ptr = 5**

B.show(); **//print: 5**

B.derived::show(); **//error, base object doesn't have derived**

B.F(15); **//\*ptr = 15**

B.show(); **//print: 15**

derived D(10); **//num = 10, \*ptr = 0**

D.show(); **//print: 10**

D.F(8); **//num = 8**

D.show(); **//print = 8**

D.base::show(); **//print = 0**

base \*p = new derived(30); **//num = 30, \*ptr = 0**

p->F(20); **//derived F runs, num = 20**

p->show(); **//base show runs(no virtual in base), print: 0**

delete p;

p = new base(40); **//\*ptr = 40**

p->F(50) **//base F runs, \* ptr = 50**

p->show(); **//base show runs, print: 50**

delete p;

}

Given the code from main above,

a. If there is any error, explain what the error is

b. Show what is printed to screen

c. For the pointer p:

What type of object is p first pointing to? **//derived**

How many constructors are run to create this object? **//base then derived**

How many destructors run at the first delete p? **//derived, then base**

3. You have the following 2 classes, which are partially written: a point base class and a TDpoint (for 3D point) derived class.

Add the function prototypes/definitions to the 2 classes for main to work as written. Don’t add extra functions that will not be needed.

class point

{

point(int x = 1, int y = 1) {this->x = x; this->y = y;}

**virtual** void show() const { cout << x << “, ” << y; }

**//Protected:**

int x, y; // x, y coordinates

};

class TDpoint : public point

{

**Tdpoint(int x = 0, int y = 0, int z = 0) : point(x, y), z(z){}**

**//Tdpoint(int x = 0, int y = 0, int z = 0) : x(x), y(y), z(z){} (must be protected x,y for this to work)**

**void show() const**

**{**

**print::show(); //print x and y**

**cout << z << endl;**

**}**

**Tdpoint operator\* (int)**

**{**

**Tdpoint temp = \*this; // assigning temp the values of A**

**temp.x \*= 3;**

**temp.y = y \* 3;**

**temp.z = z \* 3;**

**return temp;**

**}**

**protected:**

int z; // z coordinate

};

void main()

{

TDpoint A(1,2,3);

A.show; // print all 3 data fields

TDpoint C = A \* 3; // multiply x, y, z by 3 (3, 6, 9)

point D = A // D has A's x, y values

display(D); // display x,y of D

display(C); // display x,y,z of C

}

**void display(point & object)**

**{**

**object.show();**

**}**