1. T/F (8 of these)

a. A class declaration takes up space in memory

**-F. the class declaration is in the .file and is only a blueprint**

b. The default access specification for class members is private

**-T.**

c. The function definitions of member functions are stored in the class header file

**-F.**

**the only function that have function definitions in the .h file are inline function**

d. A class can only have one copy constructor

**-T. because the input argument for a copy constructor is const ClassName &**

**and that's the only choice, so no overload is possible**

e. It is possible to overload the + operator with a function that has the following function header: Class operator+ (const Class & a, const Class & b);

**-F. because the operator+ has 1 input argument, which is the right side operand of the +**

2. Reading code

Given the following function declarations for the class Num:

class Num

{

private:

int n;

public:

Num(int i = 0) {n = i;} **//constructor, can be default constructor**

Num(const Num & in){n = in.n;} **//copy constructor**

Num operator+ (const Num & in) {this->n += in.n; return \*this;}

**//operator + with Num**

Num operator+ (int i) { Num temp(i); return temp;}

**//operator + with int**

Num operator= (const Num & in) {return \*this;}

**//operator =**

};

Show which of the member functions will be used for each line of code:

1. Num one; **//default constructor**
2. Num two = one; **//copy constructor**
3. Num four = one + two; **//+ with num, copy constructor**
4. one = one + 5; **//+ with int, constructor, copy constructor,**

**= operator, copy constructor**

3. Writing code

Given a class Circle:

class Circle

{

private:

double radius;

int centerX;

int centerY;

public:

**Circle(int x = 0, int y = 0, double radius =1) : centerX(x), centerY(y), radius(radius) {}**

**double getArea() { return 3.14 \* radius \* radius; }**

**Circle operator \* (int n);**

**Void print() { cout << x << “,” << y << “,” << “radius:“ << radius << endl;}**

};

Circle::Circle operator \* (int n)

{

Circle temp;

temp.radius = radius \* n;

return temp;

//or

//Circle temp(centerX, centerY, radius \* n);

//return temp;

}

Add any member data or function such that the following can be done in main

Circle c1; // c1 is centered at (0,0) and has radius 1

cout << c1.getArea(); // prints area of c1

c1 = c1 \* 5; // radius is 5 times larger

Circle c2(5,1,7); // c2 is centered at (5,1) and has radius 7

c1 = c2; // c1 is a copy of c2

c1.print(); // prints x, y, and radius of c1