**1.** Consider the following class declaration:

class Thing

{ private: int x; int y; static int z;

public:

Thing() {x=y=z;}

static void putThing(int a) {x=a;}

};

int Thing::z=0;

Assume a program containing the class declaration defines three Thing objects with the following statements:

Thing one, two, three;

(1) How many separate instances of the x member exists?

3

(2) How many separate instances of the y member exists?

3

1. How many separate instances of the z member exists?

1

1. What value will be stored in the x and y members of each object?

0

1. Write a statement that will call the putThing member function *before* the Thing objects are defined.

Thing::putThing(1);

1. Describe the difference between making a class a member of another class (object composition) and making a class a friend of another class.

In object composition one object is nested inside another object such as:

class School {

string name;

int size;

int entrancePin;

class Student {

string firstName;

string lastName;

int grade;

};

};

In the case of a class being a friend of another class, no nesting is involved. If class A is a friend of class B, all of the member functions of A have access to B’s members, even if they are private. An example would be:

class School

{

// Declare a friend class

friend class Student;

public:

void printMember()

{

cout << entrancePin << endl;

}

private:

string name;

int size;

int entrancePin;

};

class Student

{

public:

void change(School& yourSchool, int x)

{

yourSchool.entrancePin = x; // I can access School’s private data member

}

private:

string firstName;

string lastName;

int grade;

};

1. The following class declaration has errors. Locate as many as you can and fix them.

class Circle

{ private: double diameter; int cetnerX; int cetnerY;

public:

Circle(double d, int x, int y)(diameter=d; centerX=x; centerY=y;}

//Overloaded = operator

void Circle=(Circle &right) { diameter=right.diamter; centerX=right.centerX; centerY=right.centerY;}

//… Other member functions follow ….

};

class Circle

{

private:

double diameter;

int centerX;

int centerY;

public:

Circle(double d, int x, int y)

{

diameter = d;

centerX = x;

centerY = y;

}

Circle(Circle &right)

{

diameter = right.diameter;

centerX = right.centerX;

centerY = right.centerY;

}

//… Other member functions follow ….

};

1. Complete the following tables by filling in private, protected, public or inaccessible in the right-hand columns.

(1)

|  |  |
| --- | --- |
| In a private base class, this base class  MEMBER access specification… | … becomes this access specification in the derived class. |
| private | inaccessible |
| protected | public |
| public | public |

(2)

|  |  |
| --- | --- |
| In a protected base class, this base class  MEMBER access specification… | … becomes this access specification in the derived class. |
| private | inaccessible |
| protected | inaccessible |
| public | inaccessible |

(3)

|  |  |
| --- | --- |
| In a public base class, this base class  MEMBER access specification… | … becomes this access specification in the derived class. |
| private | inaccessible |
| protected | public |
| public | public |

1. Write a function whose prototype is

char lastChar(const char \*str)

that takes a nonempty C-string as parameter and returns the last character in the string. For example, the call lastChar(“abc”) will return the character c.

(Hint) Use pointer operators.

char lastChar(const char \*str)

{

int lastCharIndex = strlen(str) - 1;

return str[lastCharIndex];

}