

1. Circle the correct answer, true or false, for the following questions:

- True** **False** The three basic principles of Object-Oriented Design are encapsulation, inheritance and polymorphism.
- True** **False** Inheritance is an "is-a" relation.
- True** **False** C++ does not support multiple inheritance (the ability for a class to be derived from 2 or more base classes).
- True** **False** Private members of a base class are private to the base class. The derived class cannot directly access them.
- True** **False** Classes may not be data members of other classes.
- True** **False** A derived class can not redefine the function members of a base class.
- True** **False** The public members of a base class can be inherited either as public or private by the derived class.
- True** **False** A pointer is a variable that contains the address of another variable as its value.
- True** **False** Increment (++) and decrement (--) operators are allowed on pointer variables.
- True** **False** The address of operator (&) returns the value pointed to by a pointer.
- True** **False** If p is a pointer variable, then the statement `p = p * 2;` is valid in C++.
- True** **False** In C++, pointer variables are declared using the reserved word pointer.

2. In 2 or 3 sentences, explain the difference between the private and protected members of a class.

Private members are private, even for classes derived from the class. Protected members are private to clients of a class, but are accessible (e.g. seem like they are public) to derived classes.

3. Consider the following statements:

```
class yClass
{
    public:
        void one();
        void two(int, int);
        yClass();
    private:
        int a;
        int b;
};
```

```
class xClass: public yClass
{
    public:
        void one();
        xClass();
    private:
        int z;
}
```

```
yClass y;
xClass x;
```

- a. The private members of yClass are public members of xClass. True or False?
- b. Mark the following statements as valid or invalid. If a statement is invalid, explain why.

(i) **valid**

```
void yClass::one()
{
    cout << a+b << endl;
}
```

(ii) **invalid, members a & b are private**

```
y.a = 15;
x.b = 30;
```

(iii) **invalid, a&b are private, not accessible to derived xClass**

```
void xClass::one()
{
    a = 10;
    b = 15;
    z = 30;
    cout << a + b + z << endl;
}
```

(iv) **invalid, same as ii, a & b are private**

```
cout << y.a << " " << y.b << " " << x.z << endl;
```

(-still using the class declarations for problem #3, previous page-)

- c. Write the definition of the default constructor of yClass so that the private data members of yClass are initialized to 0.

```
yClass::yClass()
{
    a = 0;
    b = 0;
}
```

- d. Write the definition of the default constructor of xClass so that the private data members of xClass are initialized to 0.

```
xClass::xClass()
{
    z = 0;
}
```

- e. Write the definition of the member function two of yClass so that the private data member a is initialized to the value of the first parameter of two, and the private data member b is initialized to the value of the second parameter of two.

```
void yClass::two(int p1, int p2)
{
    a = p1;
    b = p2;
}
```

4. Given the declarations:

```
int x;  
int *p;  
int *q;
```

Mark the following statements as valid or invalid. If a statement is invalid, explain why.

- a. `p = q;` valid
- b. `*p = 56;` valid (or invalid if you say that `p` and `q` have not been initialized yet)
- c. `p = x;` invalid, always invalid to assign something of type `int` to something of type `int*`
- d. `*p = *q;` valid (or invalid again if you assume `p` & `q` have not been initialized so they point to random garbage)
- e. `q = &x;` valid, always valid to assign a pointer to point to an address of a variable of the correct type
- f. `*p = q;` invalid, always invalid to assign an integer (dereferencing `p` using `*p` means it is an integer value) to have an address (`q` contents are a memory address, whatever it is pointing to)

5. What is the output of the following C++ code?

```
int x;  
int y;  
int *p = &x;  
int *q = &y;  
*p = 35;  
*q = 98;  
*p = *q;  
  
cout << x << " " << y << endl;  
cout << *p << " " << *q << endl;
```

```
98 98  
98 98
```

6. What is the output of the following C++ code?

```
int x;
int *p;
int *q;
p = new int[10];
q = p;
*p = 4;

for (int j=0; j<10; j++)
{
    x = *p;
    p++;
    *p = x + j;
}

for (int k=0; k < 10; k++)
{
    cout << *q << " ";
    q++;
}
```

4 4 5 7 10 14 19 25 32 40