

Name:

1.124 Quiz 1

Thursday October 5, 2000

Time: 1 hour 15 minutes

Answer all questions. All questions carry equal marks.

```
#include <iostream.h>
```

```
class Ball {
```

```
private:
```

```
    const float pi;
```

```
    int radius;
```

```
public:
```

```
    Ball(int r=1) {
```

```
        radius = r;
```

```
    }
```

```
    void set_radius(int radius);
```

```
    const Ball& operator=(const Ball& b);
```

```
    static int count;
```

```
    virtual void print() {
```

```
        cout << radius << endl;
```

```
    }
```

```
};
```

```
int Ball::count = 0;
```

```
class BuckyBall: public Ball {
```

```
private:
```

```
    int color;
```

```
public:
```

```
    BuckyBall(int radius, int c) {
```

```
        color = c;
```

```
    }
```

```
    void print() {
```

```
        cout << color << endl;
```

```
    }
```

```
};
```

Question 1. Show how you would initialize the member *pi* in class *Ball*.

Answer:

Use an initialization list in the constructor:

```
Ball(int r=1) : pi(3.14159f) {  
    radius = r;  
}
```

Question 2. Write the copy constructor for class *Ball*.

Answer:

Within the *public* part of the *Ball* class declaration:

```
Ball(const Ball& b) : pi(b.pi) {  
    radius = b.radius;  
}
```

Question 3. Show how you would overload the *+=* operator, so that the following code increments the radius of *b* by 2.

```
Ball b;  
b += 2;
```

Answer:

Within the *public* part of the *Ball* class declaration:

```
void operator+=(int i) {  
    radius += i;  
}
```

Question 4. Complete the definition of the member function *set_radius()*.

```
void Ball::set_radius(int radius) {
```

Answer:

```
    this->radius = radius;
```

```
}
```

Question 5. What should the = operator return so that the code

```
    Ball a, b(2), c(3);  
    a = b = c;
```

behaves as expected? Explain your answer.

```
const Ball& Ball::operator=(const Ball& b) {  
    radius = b.radius;
```

Answer:

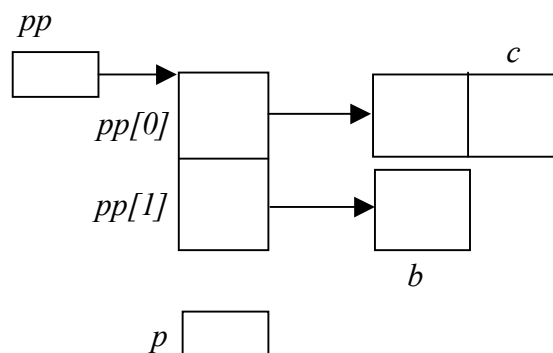
```
    return *this;
```

```
}
```

Question 6. Draw a clear diagram to illustrate the memory allocated by the following code. Label all variables on your diagram.

```
Ball b;  
Ball *p;  
Ball **pp;  
  
pp = new Ball*[2];  
pp[0] = new Ball[2];  
pp[1] = &b;  
Ball& c = pp[0][1];
```

Answer:



Question 7. How you would release the memory allocated in Question 6?

Answer:

```
delete[] pp[0];  
delete[] pp;
```

Question 8. What will be the output from the following program?

```
int count = 5;
```

```
void draw(Ball *p, int n) {  
    static int count = n;  
    cout << count << endl;  
}
```

```
void main() {  
    const int count = 2;  
    Ball b[count];  
    draw(b,7);  
    draw(b,8);  
    cout << b[1].count << count << ::count << Ball::count << endl;  
}
```

Answer:

```
7  
7  
0250
```

Question 9. Show how you would modify the *BuckyBall* constructor so that it correctly initializes the *Ball* part of a *BuckyBall* object.

Answer:

```
BuckyBall(int radius, int c) : Ball(radius) {  
    color = c;  
}
```

Question 10. What statements would you use to print out

- (i) The color of object *a*?
- (ii) The color of object *b*?
- (iii) The radius of object *b*?
- (iv) The radius of object *c*?

```
BuckyBall a(1,2);  
Ball& b = a;  
BuckyBall& c = a;
```

Answer:

```
a.print();  
b.print();  
b.Ball::print();  
c.Ball::print();
```

Question 11. What is a *protected member*? Give examples of how such a member can and cannot be used.

Answer:

A protected member of a class is a member variable or function, which can only be accessed within the definition of the class and the definitions of derived classes. e.g.

```
class Base {
protected:
    int a;
};

class Derived : public Base {
public:
    void set(int i) {
        a = i;          // Example of valid usage.
    }
};

void main() {
    Derived x;

    x.a = 7;            // Illegal.
}
```

Question 12. Give the definitions of the destructors for the *Ball* and *BuckyBall* classes.

Answer:

In the public part of the declaration for class *Ball*:

```
virtual ~Ball() {}
```

In the public part for the declaration for class *BuckyBall*:

```
~BuckyBall() {}
```