

**COSC 1305: Midterm Exam (S)**

**Name:** \_\_\_\_\_

D. Mirkovic, Spring 2005

1. (20) Answer the following questions

a) (5) Which language is more strongly typed: C or C++?

*The C++ language is more strongly typed.*

b) (5) Can procedural programming be done in C++?

*Yes, C++ supports procedural programming, but there are few advantages to using C++ in this way.*

c) (5) Code a statement that outputs three values, of type int, long, and float, to a stream.

```
int n;  
long z;  
float f;  
cout << n << z << f;
```

d) (5) Code a statement that lets the user input two long integers.

```
long l1, l2;  
cin >> l1 >> l2;
```

e) (5) Define encapsulation.

*Encapsulation is the enclosing of details inside an outer shell.*

2. (30) Consider a catalog application which represents the information about products a company sells.

a) (10) Create a class for the products in the catalog. Each Product should contain the following data members

```
long id;           // ID number of the product  
long quantity;    // Quantity on hand  
float price;       //Price of the product
```

The class should contain a default constructor, a constructor that initializes all data members, constant member functions that return the values of data members, and a function that displays a Product object on the screen.

```

class Product {
public:
    Product();
    Product(long id, long quantity, float price);
    long GetId() const;
    long GetQuantity() const;
    float GetPrice() const;
    void Display() const;
private:
    long id;           // ID number of the product
    long quantity;    // Quantity on hand
    float price;      //Price of the product
};

```

b) (10) Write an implementation for the Product class containing all member functions.

```

#include "product.h"

Product::Product()
{
    id = 0;
}
Product::Product( long idVal, long quantityVal, float priceVal )
{
    id = idVal;
    quantity = quantityVal;
    price = priceVal;
}
long Product::GetId() const
{
    return id;
}
long Product::GetQuantity() const
{
    return quantity;
}
float Product::GetPrice() const
{
    return price;
}
void Product::Display() const
{
    cout
        << "ID:          " << id << '\n'
        << "Quantity: " << quantity << '\n'
        << "Price:      " << price << endl;
}

```

- c) (10) Write a short test program that creates and displays several Product objects.

```
#include "product.h"
int main()
{
    Product P(10000, 34, 50.25f);
    P.Display();
    cout << "Price: " << P.GetPrice() << "\n";
    cout << "Quantity: " << P.GetQuantity() << "\n";
    cout << "Creating a second product...\n";
    Product Q(20000, 44, 150.75f);
    Q.Display();
    cout << "Product ID: " << P.GetID() << endl;

    return 0;
}
```

3. (20) Answer the following questions

- a) (5) How are the following two statements interpreted differently by the compiler?

```
#define WinHeight 500
const unsigned WinHeight = 500;
```

*The first is a macro, which simply substitutes the string "500" into the program's source code at each point that it encounters the name "WinHeight". The second example is a constant variable declaration; when this variable appears in the program's source code, the compiler will verify its data type and make sure its use conforms to the syntax and semantics of C++.*

- b) (5) Why do inline functions help to speed up a program?  
*They eliminate the overhead of the code required for calling a function and returning from the function.*
- c) (5) Is a copy of each member function stored in each instance of a class?  
*No.*
- d) (5) What is a const-qualified member function?  
*A function that is guaranteed not to modify the current class object.*

4. (30) Answer the following questions

- a) (5) When are friend functions necessary?  
*They are necessary when a nonmember function needs direct access to the private and protected members of a class.*
- b) (5) Can a non-member function declare itself a friend of another class?  
*No, only the class can do that.*
- c) (5) Why can't a class destructor be overloaded?  
*Because the destructor's name cannot change, and it has no parameters: It would be impossible for two destructors to have different signatures.*
- d) (5) Can a constant value be passed to a function that has a reference parameter?  
*No, because the parameter must be a modifiable lvalue.*
- e) (10) Modify the `Product` class defined in Problem 2. Overload the `<<` operator for stream output and replace the `Display` member function with it.

```
class Product {
public:
    Product();
    Product(long id, long quantity, float price);
    long GetId() const;
    long GetQuantity() const;
    float GetPrice() const;
    friend ostream & operator <<( ostream & os, const Product & P );
private:
    long id;           // ID number of the product
    long quantity;    // Quantity on hand
    float price;      //Price of the product
};

ostream & operator <<( ostream & os, const Product & P )
{
    os
        << "ID:          " << P.id << '\n'
        << "Quantity: " << P.quantity << '\n'
        << "Price:       " << P.price;
    return os;
}
```