**Name: Dalia Muhammad Abdel-Rehim Qadry**

**Section: 1**

**Assignment: 1**

1. What is a class?

A class is a programming construct that defines the common state and behavior of a group of similar objects

2. How does a class accomplish abstraction, encapsulation, and data hiding?

In OO and C++, an abstraction is the simplest interface to an object that provides all the features and services the intended users expect In C++ class is used through its access control mechanism to implement the abstract interface

3. What is the relationship between an object and a class?

A class is a definition of the behavior of an object, Objects are instances of classes

4. In what way, aside from being functions, are class function members different from class data members?

Member functions can (and should) be used to interact with data contained within user defined types. User defined types provide flexibility in the "[divide and conquer](https://en.wikipedia.org/wiki/divide_and_conquer)" scheme in program writing. In other words, one programmer can write a user defined type and guarantee an interface. Another programmer can write the main program with that expected interface. The two pieces are put together and compiled for usage. User defined types provide encapsulation defined in the Object Oriented Programming (OOP) paradigm.

Within classes, to protect the data members, the programmer can define functions to perform the operations on those data members. Member functions and functions are names used interchangeably in reference to classes. Function prototypes are declared within the class definition. These prototypes can take the form of non-class functions as well as class suitable prototypes. Functions can be declared and defined within the class definition.

6. When are class constructors called? When are class destructors called?

A default constructor is a constructor that is used to create an object when you don’t provide explicit initialization values

Destructors are used to release any resources allocated by the object. e.g., class Lock might lock a semaphore, and the destructor will release that semaphore. The most common example is when the constructor uses new, and the destructor uses delete.

8. What is a default constructor? What is the advantage of having one?

A default constructor is a constructor that is used to create an object when you don’t provide

explicit initialization values.That is, it’s a constructor used for declarations.

9. Modify the Stock class definition (the version in stock20.h) so that it has member functions that return the values of the individual data members.

Note: A member that returns the company name should not provide a weapon for altering the array. That is, it can’t simply return a string reference. It could return a const reference.

// stock30.h

#ifndef STOCK30\_H\_

#define STOCK30\_H\_

class Stock

{

private:

std::string company;

long shares;

double share\_val;

double total\_val;

void set\_tot() { total\_val = shares \* share\_val; }

public:

Stock(); // default constructor

Stock(const std::string & co, long n, double pr);

~Stock() {} // do-nothing destructor

void buy(long num, double price);

void sell(long num, double price);

void update(double price);

void show() const;

const Stock & topval(const Stock & s) const;

int numshares() const { return shares; }

double shareval() const { return share\_val; }

double totalval() const { return total\_val; }

const string & co\_name() const { return company; }

};

10. What are this and \*this?

The this pointer is available to class methods. It points to the object used to invoke the method. Thus, this is the address of the object, and \*this represents the object itself.

Programming Exercises

1. Provide method definitions for the class described in Chapter Review

Question 5

and write a short program that illustrates all the features.

class definition file and a function definition for the

methods file:

// prototype

Stonewt operator\*(double mult);

// definition — let constructor do the work

Stonewt Stonewt::operator\*(double mult)

{

return Stonewt(mult \* pounds);

}

2. Here is a rather simple class definition:

class Person {

private:

static const LIMIT = 25;

string lname; // Person’s last name

char fname[LIMIT]; // Person’s first name

public:

Person() {lname = ""; fname[0] = ‘\0’; } // #1

Person(const string & ln, const char \* fn =

"Heyyou"); // #2

// the following methods display lname and fname

void Show() const; // firstname lastname format

void FormalShow() const; // lastname, firstname

format

};

(It uses both a string object and a character array so that you can compare how the two forms are used.) Write a program that completes the implementation by providing code for the undefined methods. The program in which you use the class should also use the three possible constructor calls (no arguments, one argument, and two arguments) and the two display methods. Here’s an example that uses the constructors and methods:

Person one; // use default constructor

Person two("Smythecraft"); // use #2 with one default argument

Person three("Dimwiddy", "Sam"); // use #2, no defaults

one.Show();

cout << endl;

one.FormalShow();

// etc. for two and three

A member function is part of a class definition and is invoked by a particular object. The member function can access members of the invoking object implicitly, without using the membership operator. A friend function is not part of a class, so

it’s called as a straight function call. It can’t access class members implicitly, so it

must use the membership operator applied to an object passed as an argument.