EECE.3220: Data Structures

Key Questions

ADTs and Classes (Lectures 6-8)

QUESTIONS

1. Explain what an abstract data type (ADT) is.
2. Explain what classes and objects are.
3. Explain what data members are and how they are typically accessed.
4. Show the general syntax of a class definition and explain the difference between private and public members of a class.
5. Explain the purpose of constructors.
6. Explain the difference between default and parameterized constructors. Write the default and parameterized constructors for the GradeBook class.
7. Explain function overloading and default arguments in C++.
8. Describe the different types of class relationships. Focus on composition, as demonstrated by the Point and Rectangle classes defined near the end of this handout.
9. Describe how data members of an object must be set if that object is inside another object.

EXAMPLES

1. Explain the example code below, which provides the class definition and function definitions for a simple class with one data member.

GradeBook.h

```
// GradeBook class interface
class GradeBook
{
public:
   // function that sets the course name
   void setCourseName( string name );

   // function that gets the course name
   string getCourseName();

   // function that displays a welcome message
   void displayMessage();

private:
   string courseName; // course name for this GradeBook
};
```

GradeBook.cpp

```
// GradeBook class implementation
#include "GradeBook.h"

// function that sets the course name
void GradeBook::setCourseName( string name ) {
      courseName = name;
}

// function that gets the course name
string GradeBook::getCourseName() {
      return courseName;
}

// function that displays a welcome message
void GradeBook::displayMessage() {
     cout << "Welcome to the grade book for\n" << courseName
          << "!" << endl;
}
```

2. The example program below shows how objects are declared and their member functions are called. Use the space below the program for notes on these topics.

```
int main()
{
   string nameOfCourse;  // string of chars to store course name
   GradeBook myGradeBook; // new GradeBook object

   // display initial value of courseName
   cout << "Initial course name is: "
          << myGradeBook.getCourseName() << endl;

   // prompt for, input and set course name
   cout << "\nPlease enter the course name:" << endl;
   getline( cin, nameOfCourse ); // read course name with blanks

   myGradeBook.setCourseName( nameOfCourse );
   cout << endl;
   myGradeBook.displayMessage();
   return 0;
}
```

3. Assume that GradeBook.h is as follows:

```
#include <string>
using std::string;
class GradeBook
{
public:
    GradeBook( );
    GradeBook( string name );
    void setCourseName(string name);
  string getCourseName();
    void displayMessage();
private:
   string name;
}; // end class GradeBook
```

Which of the following statements would be legal in a main program that uses the GradeBook class? Which would cause compiler errors? What code could we use to fix those errors?

a. `GradeBook g1(3220);`

b. `GradeBook g2;`

c. `setCourseName(g2);`

d. `g2.courseName = "EECE.3220";`

e. `string s = g2.getCourseName();`

f. `g2.displayMessage;`

Questions 4-7 refer to the following class definitions:

```
// Point definition, taken from Point.h
class Point {
public:
   Point();                     // Default constructor
   Point(double X, double Y);  // Parameterized constructor
   void setX(double newX);     // Set X coordinate
   void setY(double newY);     // Set Y coordinate
   double getX();              // Returns X coordinate
   double getY();              // Returns Y coordinate
   void printPoint(ostream &out);   // Output Point as
                                    //  (xCoord, yCoord)
private:
   double xCoord;         // X coordinate
   double yCoord;         // Y coordinate
};

// Rectangle definition, taken from Rectangle.h
class Rectangle {
public:
   Rectangle();                // Default constructor
   // Parameterized constructor
   Rectangle(double h, double w, double x, double y);
   double getHeight();    // Return height
   double getWidth();     // Return width
   Point getOrigin();     // Return origin
   void setHeight(double h);   // Change height
   void setWidth(double w);    // Change width
   void setOrigin(Point p);    // Change origin
   double area();    // Return area of rectangle
private:
   double width;
   double height;
   Point origin;     // Lower left corner
};
```

4. Write code for each of the following functions:

a. `Point Rectangle::getOrigin() {`

`}`

b. `void Rectangle::setOrigin(Point p) {`

`}`

5. Describe the purpose of an initialization list.

6. Rewrite the `Rectangle` default constructor to use an initialization list.

7. Write a parameterized constructor for the `Rectangle` class that takes four arguments: height (`h`), width (`w`), and the X and Y coordinates of the origin (`x`, `y`).