Activity #21: Constructors and Overloading
Recorder's Report

Manager:	Reader:
Recorder:	Driver:
Date:	Score: Satisfactory / Not Satisfactory

Record your team's answers to the key questions (marked with 🔑) below.

a) Model 1, Question #4

b) Model 2, Question #12

c) Model 3, Question #15

Activity #21: Constructors and Overloading

In this activity, you will work in teams of 3–4 students to learn new concepts. This activity will introduce you to constructors and operator overloading in C++.

Content Learning Objectives

After completing this activity, students should be able to:

- Explain how constructors are defined and called
- Explain how the implicit parameter `this` functions in C++
- Explain what it means to overload an operator in the context of a class

Process Skill Goals

During the activity, students should make progress toward:

- Write constructors for a given class
- Write operator methods for a given class

Model 1 A C++ Class for a 2×2 Matrix

```
 1  class Matrix {
 2    public:
 3      Matrix();
 4      void setValues(int,int,int,int);
 5      void print();
 6      int getTrace();
 7      int getDeterminant();
 8    private:
 9      int a,b,c,d; // Entries in | a b |
10                   // matrix     | c d |
11  };
12
```

```
 1  Matrix::Matrix() {
 2    a = 1;  b = 0;
 3    c = 0;  d = 1;
 4  }
 5
 6  int main() {
 7    Matrix A,B;
 8    A.setValues(2,2,2,2);
 9    A.print();
10    B.print();
11  }
12
```

Refer to Model 1 above as your group develops consensus answers to the questions below.

Questions (15 min) Start time:

1. The C++ code snippets above define a class for a 2×2 matrix of integers and given an example main program. Recall that a 2×2 matrix is a grid of two rows and two columns. Write the matrix that corresponds to the given values for `a`, `b`, `c`, and `d` below. The first one is done for you.

a) `a=1, b=2, c=3, d=4`

c) `a=0, b=1, c=1, d=0`

b) `a=1, b=1, c=1, d=1`

d) `a=1, b=0, c=0, d=1`

2. Without running any code, predict the output produced by the following statements in the `main` program of this model.

a) `A.print();` on line 29

b) `B.print();` on line 30

3. The complete code for this model can be found in `activity21a.cpp`. Run the code. Were your predictions accurate?

4. The method `B.setValues()` was never called in this code. Explain how the values of the entries for matrix `B` were initialized.

5. A *constructor* for a class is a method that is called automatically when a new object variable of the class is created. Perform the following tasks to help determine how a constructor is defined in C++.

a) Change the name of the method `Matrix()` to `matrix()` (with a lower case 'm') on lines 8 and 19 of the model. What happens when you compile the program?

b) Add the appropriate function type on lines 8 and 19 so that the program compiles. Run it several times and observe how the output different from the original model.

c) Based on your observations above, how does C++ determine if an object method is a constructor?

Model 2 More Constructor Options

```
int main() {
  Matrix Z(0);       // create a matrix of all zeros
  cout << "Z = ";
  Z.print();

  Matrix I;          // create identity matrix [ [1,0], [0,1] ]
  cout << "I = ";
  I.print();

  Matrix A(1,2,3,4); // create matrix [ [1,2], [3,4] ]
  cout << "A = ";
  A.print();
}

```

Refer to Model 2 above as your group develops consensus answers to the questions below.

Questions (15 min) Start time:

6. Based on the model (with comments) above, what output would you expect this program to produce?

7. The file `activity21b.cpp` contains the same class definition and methods as in model 1, but with the `main` program above. Try compiling the program and explain the errors you see.

8. Thinking back to what you learned about functions in CPTR 141 (or some other prerequisite class), answer the following questions.

a) What does it mean to *overload* a function name?

b) What is the *signature* of a function?

c) Based on the error messages you saw above, what are the signatures for the missing constructors in this model?

9. Suppose we wish to *overload* the constructor for this class to allow for initializing all matrix entries to a single value (as seen on line 27 of this model). We can accomplish this by adding the following prototype to the public portion of the `Matrix` class.

```
Matrix(int);
```

Add this prototype to the class in `activity21b.cpp` and then define it similarly to how the original constructor was defined on lines 19-24 of the first model. Hint: comment out appropriate lines in the `main` program so that you can test your code.

10. Give a method prototype for the constructor needed to initialize matrix A in this model. Where should you put this prototype?

11. Fill in the definition of this constructor below.

```
Matrix::Matrix(int a, int b, int c, int d);
```

12. Add this constructor to the code in `activity21b.cpp` and then compile and run the code (uncommenting lines if `main` if needed). Does it work as expected?

13. In the example above, the parameter names (`a`, `b`, `c`, and `d`) *shadow* the data members of the same name. You can still access the class data members using the *implicit parameter* `this` to indicate that you want the data members of the class. For example, one line of your definition above might be:

```
this->a = a;
```

Adjust your definition to use `this` so that the constructor works as expected.

Model 3 Adding Matrices

```
 1 class Matrix {
 2   public:
 3     Matrix();
 4     Matrix(int);
 5     Matrix(int,int,int,int);
 6     void setValues(int,int,int,int);
 7     void print();
 8     int getTrace();
 9     int getDeterminant();
10     Matrix operator+(Matrix);
11   private:
12     int a,b,c,d; // Entries in | a b |
13                  // matrix     | c d |
14 };
15
```

```
 1 Matrix Matrix::operator+(Matrix rhs) {
 2   Matrix Sum;
 3   Sum.a = a + rhs.a;
 4   Sum.b = b + rhs.b;
 5   Sum.c = c + rhs.c;
 6   Sum.d = d + rhs.d;
 7   return Sum;
 8 }
 9
10 int main() {
11   Matrix A(2),B(1,2,3,4);
12   Matrix C = A + B;
13   C.print();
14 }
15
```

Refer to Model 3 above as your group develops consensus answers to the questions below.

Questions (20 min) Start time:

14. Recall that matrices are added together by adding their corresponding entries. So, for example:

$$2513 + 1301 = 3814$$

Use this same notation to express the matrix sum computed by the `main` program in this model.

15. Since we defined the `Matrix` class ourself, we have to tell C++ how to add two matrices together with the + operator. This is called *operator overloading*. Find the lines in the model above where each of the following is accomplished.

a) Adding a prototype function for the addition operation:

b) Defining how two `Matrix` objects are added together:

c) Adding together the two top-right entries in the matrices:

d) Adding together the two bottom-left entries in the matrices:

16. A mathematical definition of matrix multiplication is given below. Using `activity21c.cpp`, overload the `*` operator to allow you to multiply two `Matrix` objects together using the command `C = A * B;`.

$$a_1 b_1 c_1 d_1 * a_2 b_2 c_2 d_2 = a_1 * a_2 + b_1 * c_2 a_1 * b_2 + b_1 * d_2 c_1 * a_2 + d_1 * c_2 c_1 * b_2 + d_1 * d_2$$