CS 161 Self Review Exercises

Chapter Seven

7.1

// Answer each of the following:

a) Lists and tables of values can be stored in **arrays** or **vectors.**
b) An array's elements are related by the fact that they have the same **array name** and **type**.
c) The number used to refer to a particular element of an array is called its **subscript or index.**
d) A **constant variable** should be used to declare the size of an array, because it eliminates magic numbers.
e) The process of placing the elements of an array contains a particular key value is called **sorting** the array.
f) The process of determining if an array contains a particular key value is called **searching** the array.
g) An array that uses two subscripts is to as a **2-dimensional** array.

7.2

//State whether the following are true or false.

a) A given array can store many different types of values.
    i) false: each array can only have a single type of data.
b) An array subscript should normally be a data type float.
    i) false: an array subscript should normally be an integer.
c) If there are fewer initializers in an initializer list than there are elements in the array, the remaining elements are initialized to the last value in the initializer list.
    i) false: the uninitialized elements are initialized to zero.
d) It's an error if an initializer list has more initializers than there are elements in the array.
    i) true.

7.3

// Write one or more statements that perform the following tasks for an array called fractions:

a)
b)
c)
d)
e)
f)

g)
h)
i)

7.4

// Answer the following questions regarding a two-dimensional array called table:

a)
b)
c)
d)

7.5

// Find and correct the error in each of the following program segments:

a)
b)
c)
d)

Chapter Six

6.1

//Answer each of the following:

a) Program Components in C++ are called **functions** and **classes**.
b) A function is invoked with a **function call**.
c) A variable known only within the function in which it is defined is called a **local variable.**
d) The **return** statement in a called function passes the value of an expression back to the calling function.
e) The keyword **void** is used in a function header to indicate that a function does not return a value or to indicate that a function contains no parameters.
f) An identifier's **scope** is the portion of the program in which the identifier can be used.
g) The three ways to return control from a called function to a caller are **return, return expression, or encounter the closing right brace of a function.**
h) A **function prototype** allows the compiler to check the number, types, and order of the arguments passed to a function.
i) Function **rand.** is used to produce random numbers.
j) Function **srand()** is used to set the random number seed tortandomize the number sequence generated by function rand.
k)
l) A variable in a function to retain its value between calls to the function, it must be declared **global.**

m) A function that calls itself either directly or indirectly (ie., through another function) is a **recursive** function.
n) A recursive function typically has two components... one that provides a means for the recursion to terminate by testing for a **base** case and one that expresses the problem as a recursive call for a slightly simpler problem that the original call.
o) It's possible to have various functions with the same name that operate on different types or numbers of arguments. This is called function **overloading.**
p) The **:: unary scope** enables access to a global variable with the same name as a variable in the current scope.
q) The **const.** qualifier is used to declare read-only variables.
r) A function **template** enables a single function to be defined to perform a task on many different data types.

6.2

// For the program in figure 6.30, state the scope (global namespace scope or block scope) of each of the following elements:

a) The variable x in main **is a block scope.**
b) The variable y in function cube's definition is **a block scope.**
c) The function cube is **in the global namespace.**
d) The function main is **in the global namespace.**
e) The function prototype is in the **global namespace.**

6.3

//Write a program that tests whether the examples of the math library function calls shown in fig 6.2 actually produce the indicated results.

```
#include <iostream>
#include <cmath>
using namespace std;

int main() {
   cout << "ceil(9.2) is " << ceil(9.2) << " (10.0)"
      << "\ncos(0.0 is " << cos(0.0) << " (1.0)"
      << "\nexp(1.0) is " << exp(1.0) << " (2.718282)"
      << "\nfabs(-8.76) is " << fabs(-8.76) << " (8.76)"
      << "\nfloor(9.2) is " << floor(9.2) << " (9.0)"
      << "\nfmod(2.6, 1.2) is " << fmod(2.6, 1.2) << " (0.2)"
      << "\nlog(2.718282) is " << log(2.718282) << " (1.0)"
      << "\nlog10(10.0) is " << log10(10.0) << " (1.0)"
      << "\npow(2, 7) is " << pow(2, 7) << " (128)"
      << "\nsin(0.0) is " << sin(0.0) << " (0.0)"
      << "\nsqrt(9.0) is " << sqrt(9.0) << " (3.0)"
      << "\ntan(0.0) is " << tan(0.0) << " (0)" << endl;
}
```

6.4

// Give the function header for each of the following functions:

a) Function hypotenuse that takes two double precision, floating point arguments, side1 and side2, and returns a double precision, floating point result.

```
double hypotenuse( double side1, double side2) { }
```

b) Function smallest that takes three integers, x, y, and z, and returns an integer.

```
int smallest( int x, int y, int z) { }
```

c) Function instructions that does not receive any arguments and does not return a value.

```
void instructions() { }
```

d) Function intToDouble that takes an integer argument, number, and returns a double precision, floating point result.

```
double intToDouble (int number) { }
```

6.5

// Give the function prototype (without parameter names) for each of the following:

a) double hypotenuse(double, double)
b) int smallest(int, int, int)
c) void instructions()
d) double intToDouble(int)

6.6

// Write a declaration for double-precision, floating-point variable lastVal that should retain its value between calls to the function in which its defined.

```
static double lastVal{};
```

6.8

// Why would a function prototype contain a parameter type declaration such as a double&?

This creates a reference parameter of type “reference to double” that enables the function to modify the original variable in calling the function.

Chapter Five

5.1

//Fill in the blanks for each of the following statements:

a) Typically, **for** statements are used for counter-controlled iteration and **while** statements for sentinel-controlled iteration.
b) The do...while statement tests the loop-continuation condition **after** executing the loop’s body; therefore, the body always executes at least once.
c) The **switch** statement selects among multiple actions based on the possible values of an integer variable or expression.
d) The **continue** statement, when executed in an iteration statement, skips the remaining statements in the loop body and proceeds with the next iteration step.

e) The "**&&"** operator can be used to ensure that two conditions are both true before choosing a certain path of execution.
f) If the loop-continuation condition in a for header is initially **false**, the program does not execute the for statement's body.

5.2

// State whether each of the following is true or false. If the answer is false, explain why.

a) The default case is required in the switch statement.
    i) FALSE: Although it is heavily suggested to ensure that there are no bugs or errors in the code, the default is not required because all of the possible statements could be covered specifically in other areas of the switch statement.
b) The break statement is required in the default case of a switch selection statement.
    i) FALSE: the function will break through the loop itself.
c) The expression (x > y&& a < b) is true if either the expression x > y is true or the a < b expression is true.
    i) FALSE: Both of them have to be true.
d) An expression containing the || operator is true if either or both of its operands are true.
    i) TRUE.

5.3

// Write a C++ statement or a set of c++ statements to accomplish each of the following

a) Sum the odd integers between 1 and 99 using a for statement. Use the unsigned int variables sum and count.

```
for (count{1}; count <=99; count = count + 2) {
        sum{0};
        sum = sum + count;
}
```

b) Print the value 33.546372 in a 15-character field with precisions of 1, 2, and 3. Print each number on the same line. Left-justify each number in its field. What three values print?

```
#include <iostream>
#include <iomanip>

using namespace std;

int main()
{
   cout << fixed << left << setprecision(1) << setw(15) << 333.546372
      << setprecision(2) << setw(15) << 333.546372
      << setprecision(3) << setw(15) << 333.546372 << endl;
}
```

c) Calculate a value of 2.5 raised to the power 3 using function pow. Print the result with a precision of 2 in a field width of 10 positions. What prints?

```
#include <iostream>
#include <cmath>
using namespace std;

int main() {
   cout << fixed << setprecision(2) << setw(10) << pow(2.5, 3) << endl;
}
```

d) Print the integers from 1 to 20 using a while loop and the unsigned int counter variable x. Print only 5 integers per line.

```
unsigned int x{1};
int main() {
   while (x <= 20) {
      if (x % 5 == 0) {
         cout << x << endl;
      } else {
         cout << x << "\t";
      }
      x++;
   }
}
```

e) Repeat exercise 3 using a for statement:

```
   for (unsigned int x{1}; x <=20; x++) {
      if (x % 5 == 0) {
         cout << x << endl;
      } else {
         cout << x << "\t";
      }
   }
```

Chapter Four

4.1

//Answer each of the following questions.

a) All programs can be written in terms of three types of control statements: **sequence** , **selection** , and **iteration** .

b) The **double** selection statement is used to execute one action when a condition is true or a different action when that condition is false.

c) Repeating a set of instructions a specific number of times is called **counter-controlled/define** iteration.
d) When it isn't known in advance how many times a set of statements will be repeated, a **sentinel/signal/flag/dummy** value can be used to determine the iteration.

4.2

// Write four different C++ statements that each ad 1 integer to variable x.

```
x = x+1;
x+=1;
x++;
++x;
```

4.3

//Write C++ statements to accomplish each of the following tasks.

a) In one statement, assign the sum of the current value of x and y to z and postincrement the value of x.

```
z = x++ + y;
```

b) Determine whether the value of the variable COUNT is greater than 10. If it is, print "COUNT is greater than 10."

```
if (count > 10) {
        cout << "Count is greater than 10" << endl;
}
```

c) Predecrement the variable x by 1, then subtract it from the variable TOTAL.

```
total -= --x;
```

d) Calculate the remainder after q is divided by DIVISOR and assign the result to q. Write this statement two different ways.

```
q %= divisor;
q = q % divisor;
```

4.4

//Write C++ statements to accomplish each of the following tasks.

a) Declare variable SUM to be of type UNSIGNED INT and initialize it to 0.

```
unsigned int sum{0};
```

b) Declare variable x to be of UNSIGNED INT and initialize it to 1.

```
unsigned int x{1};
```

c) Add variable x to variable SUM and assign the result to variable SUM.

```
sum += x;
sum = sum + x;
```

d) Print "The sum is: " followed by the value of variable SUM.

```
cout << "The sum is: " << sum << endl;
```

4.5

// Combine the statements that you wrote in Exercise 4.4 into a program that calculates and prints the sum of the integers from 1 to 10. Use the WHILE statement to loop through the calculation and increment statements. The loop should terminate when the value of x becomes 11.

```
unsigned int sum{0};
unsigned int x{1};

while (x <= 10) {
        sum += x
}

cout << “The sum is: “ << sum << endl;
```

4.6

//State the values of each of these UNSIGNED INT variables after the calculation is performed, Assume that, when each statement begins executing, all variables have the integer value 5.

a) product *= x++;

```
product = 5 * 5 + 1
```

b) quotient /= ++x;

```
quotient = 5/6
```

4.7

// Write single C++ statements or portions of statements that do the following:

a) Input UNSIGNED INT variable x with cin and >>.

```
cin >> x;
```

b) Input UNSIGNED INT variable y with cin and >>.

```
cin >> y;
```

c) Declare UNSIGNED INT variable i and initialize it to 1.

```
unsigned int i{1};
```

d) Declare UNSIGNED INT variable POWER and initialize it to 1.

```
unsigned int power{1};
```

e) Multiply variable POWER by x and assign the result to POWER.

```
power *= x;
```

f) Preincrement variable i by 1.

```
++i;
```

g) Determine whether i is less than or equal to y.

```
if (i <= y) {
```

h) Output integer variable POWER with cout and <<.

```
cout << power;
```

4.8

//Write a C++ program that uses the statements in Exercise 4.7 to calculate x raised to the y power. The program should have a WHILE iteration statement.

4.9

```
// Identify and correct the errors in each of the following:
        a) while (c <= 5) {
                product *= c;
                ++c;
        b) cin >> value;
        c) if (gender == 1) {
                cout << “Woman” << endl;
                else; {
                cout << “Man” << endl;
                }
```

4.10

```
// What is wrong with the following WHILE iteration statement?
        while (z >= 0) {
                sum += z;
        }
```

Chapter Three

3.1

// Fill in the blanks for each of the following:

a) Every class definition contains the keyword **class** followed immediately by the class’s name.
b) A class definition is typically stored in a file with the **.h** filename extension.
c) Each parameter in a function header specifies both a **data type** and a **variable name**.
d) When each object of a class maintains its own version of an attribute, the variable that represents the attribute is also known as **data member**.
e) Keyword public is a **access specifier**.
f) Return type **void** indicates that a function will perform a task but will not return any information when it completes its task.
g) Function **getLine** from the <string> library reads characters until a newline character is encountered, then copies those characters into the specified string.
h) Any file that uses a class can include the class’s header via an **#include**.

3.2

// State whether each of the following is true or false. If false, explain why.

a) By convention, function names begin with a capital letter and all subsequent words in the name begin with a capital letter.
   i) **False.** Camel capitalization, where the first word is lowercase and all subsequent words in the function name are capitalized, is the default. Functions can also use underscores after the first word instead of just using capital letters.

b) Empty parentheses following a function name in a function definition indicate that the function does not require any parameters to perform its task.
   i) **True.**

c) Data members or member functions declared with access specifier “private” are accessible to member functions in the class in which they’re declared.
   i) **True.** They are not, however, accessible to functions outside of the class in which they are declared.

d) Variables declared in the body of a particular member function are known as data members and can be used in all member functions of the class.
   i) **False.** Variables declared in a specific functions cannot be used in subsequent member functions. They are local, and therefore cannot be used outside of the local member function.

e) Every function’s body is delimited by left and right braces.
   i) **True.**

f) The types of arguments in a function call must be consistent with the types of the corresponding parameters in the function’s parameter list.
   i) **True.**

3.3

//What is the difference between a local variable and a data member?
Local variables are declared inside of member functions, and can only be used in the member function in which they are declared. Data members are declared outside of member functions to be used throughout the class.

3.4

//Explain the purpose of a function parameter. What is the difference between a parameter and an argument?
A parameter represents additional information supplied for the member function. An argument supplies values into the function call. An argument is used in a function’s parameters, but a function’s parameters cannot be called into an argument, duh.

3.5

//What is a default constructor? How are an object’s data members initialized if a class has only a default constructor defined by the compiler?

A default constructor is a constructor that is used for any class that does not explicitly define a constructor. Default constructors have no parameters. With a default constructor, the object's data members are initialized by calling the default constructor for each data member that happens to be an object of another class. (3.4)

3.6

//Explain the purpose of a data member.
A data member is a variable that is declared outside of a member function, but inside of a class. this means that a variable defined as a data member can be used in all member functions inside of its class.

3.7

//Explain how a program could use class string without inserting a using directive.
While there is a lot of room for syntax error when using classes without inserting a using directive, in order to use the class string you must type std:: before each use of the class string. ie. , std::string accountName.

3.8

//Explain why a class might provide a set function and a get function for a data member.
A class might provide a set function and a get function for a data member so that the data member can be used in different classes. A get function calls a class into another function, and the set function helps the data member execute in the other function.

Chapter Two

2.7

// Discuss the meaning of each of the following objects:

a) std::cin
   This is a command that allows the user to enter an input. It is followed by a bitwise right shift, >>, and then the name of the variable that the input will be used by.

b) std::cout
   This is a command that allows the programmer to output a message. It is followed by a bitwise left shift, <<, and then the message that the programmer wishes to display. This message could contain words, numbers, or even the value of a variable.

2.8

// Fill in the blanks in each of the following:

a) **Comments** are used to document a program and improve its readability.
b) The object used to print information on the screen is **std::cout.**
c) A C++ statement that makes a decision is **an if condition.**

d) Most calculations are normally performed by **arithmetic** statements.
e) The **#include** object inputs values from the keyboard.

2.9

//Write a single C++ statement or line that accomplished each of the following:

a) Print the message “Enter two numbers.”
    i) std::cout << “Enter two numbers.” << endl;
b) Assign the product of variables b and c to variable a:
    i) a = b * c;
c) State that a program performs a payroll calculation.
    i) //This program performs a payroll calculation.
d) Input three integer values from the keyboard into integer variables a, b, and c.
    i)

```
int a{4};
int b{23};
int c{2039};
```

2.10

// State which of the following are true and which are false. If false, explain your answers.

a) All operators are evaluated from left to right.
    i) False. For parentheses, at least, C++ standard does not specify the order in which parenthesized subexpressions are evaluated.
b) The following are all valid variable names: _under_bar_, m928134, t5, j7, her_sales, his_account_total, a, b, c, z, z2.
    i) False. Underscores cannot be used in variable names, much less at the *beginning* of a variable name.
c) The statement cout << “a = 5;”; is a typical example of an assignment statement.
    i) False. An assignment statement assigns an integer to a variable. This line of code will only physically show the words “a=5; .
d) A valid arithmetic expression with no parentheses is evaluated from left to right.
    i) True.
e) The following are all invalid variable names: 3g, 87, 67h2, h22, 2h.
    i) False. A variable name is valid if it is a combination of numbers and letters.

2.11

// Fill in the blanks for each of the following:

a) What arithmetic operations are on the same level of precedence as multiplication?
    i) Division.
b) When parentheses are nested, which set of parentheses is evaluated first in an arithmetic expression?
    i) The most outward set of parentheses.

c) A location in the computer's memory that may contain different values at various times throughout the execution of a program is called a **destructive operation.**

2.12

// What, if anything, prints when each of the following statements is performed? If nothing prints, then answer "nothing." Assume x = 2 and y = 3.

a) cout << x;
    i) 2
b) cout << x + x;
    i) 4
c) cout << "x=";
    i) x=
d) cout << "x =" << x;
    i) x = 2
e) cout << x + y << " = " << y + x;
    i) 5 = 5
f) z = x + y;
    i) nothing; to print z, you would have to use the statement "cout"
g) cin >> x >> y;
    i) nothing
h) // cout << "x + y =" << x + y;
    i) nothing; that's a comment.
i) cout << "/n";
    i) it would either print "/n" or it would print nothing because /n signals an endline.

2.13

// Which of the following statements contain variables whose values are replaced?

a) cin >> b >> c >> d >> e >> f;
    i) The values of these variables are replaced by the input. I think.
b) p = i + j + k + 7;
    i) p's value is replaced by that of i, j, k, and 7.
c) cout << "variables whose values are replaced";
    i) There are no variables in this statement, therefore there is nothing to replace.
d) cout << "a = 5";
    i) There are no variables whose values are replaced here. This is a printed statement, so there's no actual variables and no actual replacing.

2.14

// Given the algebraic equation y = ax^3 + 7, which of the following, if any, are correct C++ statements for this equation?

a) y = a * x * x * x + 7;
    i) YES
b) y = a * x * x * (x + 7);

i) NO

c) y = (a * x) * x * (x + 7);
   i) NO
d) y = (a * x) * x * x + 7;
   i) NO
e) y = a * (x * x * x) + 7;
   i) YEs
f) y = a * x * (x * x + 7);
   i) NO

2.15

// State the order of evaluation of the operators in each of the following C++ statements and show the value of x after each statement is performed.

a) x = 7 + 3 * 6 / 2 - 1
   i) First, 3 is multiplied by 6. Then, 18 (3 * 6) is divided by 2. Then, 7 plus 9 (18/2) and then minus 1. The outcome is 15
b) x = 2 % 2 + 2 * 2 - 2 / 2;
   i) First, %. Then, *. Then, /. Then, +. Then, -.
c) x = (3 * 9 * (3 + (9 * 3 / (3))));
   i) Starting with the innermost parentheses, 9 * 3. then, 27 / 3. Then, 9 + 3. Then, 3 * 9 * 12.

Chapter One

1.1

//Fill in the blanks in each of the following statements:

a) Computers process data under the control of sets of instructions called **computer programs**.
b) The key logical units in the computer are the **input unit, output unit, memory unit, Arithmetic and Logic unit (ALU), Central Processing unit (CPU), and the secondary storage unit.**
c) The three types of languages discussed in the chapter are **Machine languages, Assembly languages, and High-Level languages.**
d) The programs that translate high-level language programs into machine language are called **compilers.**
e) **Android** is an operating system for mobile devices based on the Linux kernel and Java.
f) **Release Candidates** software is generally feature complete and (supposedly) bug free and ready for use by the community.
g) The Wii remote, as well as many smartphones, uses an **accelerometer**, which allows the device to respond to motion.

1.2

//Fill in the blanks in each of the following sentences about the C++ environment.

a) C++ programs are normally typed into a computer using an **editor** program.
b) In a C++ system, a **preprocessor** program executes before the compiler's translation phase begins.
c) The **linker** program combines the output of the compiler with various library functions to produce an executable program.
d) The **loader** program transfers the executable program from disk to memory.

1.3

//Fill in the blanks in each of the following statements (based on Section 1.8):

a) Objects have the property of **information hiding:** although objects may know how to communicate with one another across well-defined interfaces, they normally are not allowed to know how other objects are implemented.
b) C++ programmers concentrate on creating **classes**, which contain data members and the member functions that manipulate those data members and provide services to clients.
c) The process of analyzing and designing a system from an object- oriented point of view is called **object-oriented analysis and design (OOAD)**.
d) With **inheritance**, new classes of objects are derived by absorbing characteristics of existing classes, then adding unique characters of their own.
e) **The Unified Modeling Language** is a graphical language that allows people who design software systems to use an industry- standard notation to represent them.
f) The size, shape, color and weight of an object are considered **attributes** of the object's class.

1.4

//Fill in the blanks in each of the following statements:

a) The logical unit of the computer that receives information from outside the computer for use by the computer is the **memory unit.**
b) The process of instructing the computer to solve a problem is called a **member-function call.**
c) Assembly language is a type of computer language that uses English-like abbreviations for machine-language instructions.
d) An **output unit** is a logical unit of the computer that sends information which has already been processed by the computer to various devices so that it may be used outside the computer.
e) The **Memory Unit** and **Secondary Storage Unit** are logical units of the computer that retain information.
f) The **Arithmetic and Logic Unit (ALU)** is a logical unit of the computer that performs calculations.
g) The **Central Processing Unit** is a logical unit of the computer that makes logical decisions.

h) **High-Level** languages are most convenient to the programmer for writing programs quickly and easily.
i) The only language a computer can directly understand is that computer's **Machine Language.**
j) The **Central Processing Unit (CPU)** is a logical unit of the computer that coordinates the activities of all the other logical units.

1.5

//Fill in the blanks in each of the following statements:

a) **PHP** initially became widely known as the development language of the UNIX operating system.
b) The **Objective-C** programming language was developed by BJarne Stroustrup in the early 1980s at Bell Laboratories.

1.6

//Fill in the blanks in each of the following statements:

a) C++ programs normally go though six phases: **Editing, Preprocessing, Compiling, Linking, Loading, and Execution.**
b) A **C++ compiler** provides many tools that support the software development process, such as editors for writing and editing programs, debuggers for locating logic errors in programs, and many other features.

1.7

A watch is an object with many attributes that are used to provide time. A watch can have many features, such as an additional date mode, perhaps, in addition to its original function. There are different classes for the parts of a watch, such as the band, which has a purpose of holding the watch to one's wrist; the cogs, which have the purpose of providing the time; and the hands of the clock, which also work to provide the time but in a different way. An alarm clock inherits a time from the user at which to sound an alarm. There are many different models for clocks: watches, as previously stated, analog clocks, clocks that use arabic numbers, clocks that sit on the wall, clocks that rest on a table.