C++ PRIMER PLUS, 5th EDITION
PROGRAMMING EXERCISES
CHAPTER 12

1. Consider the following class declaration:

```
class Cow {
    char name[20];
    char * hobby;
    double weight;
public:
    Cow();
    Cow(const char * nm, const char * ho, double wt);
    Cow(const Cow & c);
    ~Cow();
    Cow & operator==(const Cow & c);
    void ShowCow() const; // display all cow data
};
```

Provide teh implementations for this class and write a short program that uses all member functions.

2. Enhance the `String` class declaration (that is, upgrade `string1.h` to `string2.h`) by doing the following:

a. Overload the `+` operator to allow you to join two strings into one.
b. Provide a `stringlow()` member function that converts all alphabetic characters in a string to lowercase. (Don't forget the `cctype` family of character functions.)
c. Provide a `stringup()` member function that converts all alphabetic characters in a string to uppercase.
d. Provide a member function that takes a `char` argument and returns the number of times the character appears in a string.

Test your work in the following program:

```
// pe12_2.cpp
#include <iostream>
using namespace std;
#include "string2.h"
int main()
{
    String s1(" and I am a C++ student.");
    String s2 = "Please enter your name:  ";
    String s3;
    cout << s2;                  // overloaded << operator
    cin >> s3;                   // overloaded >> operator
    s2 = "My name is " + s3;     // overloaded =, + operators
    cout << s2 << "\n";
    s2 = s2 + s1;
    s2.stringup();               // converts string to uppercase
    cout << "The string\n" << s2 << "\ncontains " << s2.has('A')
         << " 'A' characters in it.\n";
    s1 = "red";  // String(const char *),
                 // then String  operator=(const String )
    String rgb[3] = { String(s1), String("green"), String("blue")};
    cout << "Enter the name of a primary color for mixing light:  ";
    String ans;
    bool success = false;
```

2. (continued)

```
while (cin >> ans)
{
    ans.stringlow();          // converts string to lowercase
    for (int i = 0; i < 3; i++)
    {
        if (ans == rgb[i])    // overloaded == operator
        {
              cout << "That's right!\n";
              success = true;
              break;
              }
        }
        if (success)
              break;
        else
              cout << "Try again!\n";
    }
    cout << "Bye\n";
    return 0;
}
```

Your output should look like this sample run:

```
Please enter your name: Fretta Farbo
My name is Fretta Farbo.
The string
MY NAME IS FRETTA FARBO AND I AM A C++ STUDENT.
contains 6 'A' characters in it.
Enter the name of a primary color for mixing light; yellow
Try again!
BLUE
That's right!
Bye
```

3. Rewrite the `Stock` class, as described in Listings 10.7 and 10.8 in chapter 10, so that it uses dynamically allocated memory directly instead of using `string` class objects to hold the stock names. Also, replace the `show()` member function with an overloaded `operator<<()` definition. Test the new definition program in Listing 10.9.

4. Consider the following variation of the `Stack` class defined in Listing 10.10:

```
// stack.h -- class declaration for the stack ADT
typedef unsigned long Item;

class Stack
{
private:
    enum MAX = 10;        // constant specific to class
    Item * pitems;        // holds stack items
    int size;             // number of elements in stack
    int top;              // index for top item of stack
public:
    Stack(int n = 10);    // creates stack with n elements
    Stack(const Stack  st);
    ~Stack();
    bool isempty() const;
    bool isfull() const;
    // push() returns false if stack already is full, true otherwise
    bool push(const Item & item); // add item to stack
    // pop() returns false if stack already is empty, true otherwise
    bool pop(Item & item); // pop top into item
    Stack  operator=(const Stack & st);
};
```

As the private members suggest, this class uses a dynamically allocated array to thold the stack items. Rewrite the methods to fit this new representation and write a program that demonstrates all the methods, including the copy constructor and assignment operator.

5. The Bank of Heather has performed a study showing that ATM customers won't wait more than one minute in line. Using the simulation from Listing 12.10, find a value for the number of customers per hour that leads to an average wait time of one minute. (Use at least a 100-hours trial period.)

6. The Bank of Heather would like to know what would happen if it added a second ATM. Modify the simulation in this chapter so that it has two queues. Assume that a customer will join the first queue if it has fewer people in it than the second queue and that the customer will joint the second queue otherwise. Again, find a value for the number of customers per hour that leads to an average wait time of one minute. (Note: This is a non-linear problem in that doubling the number of ATMs doesn't double the number of customers who can be handled per hour with a one-minute wiat maximum.)