**Instructions:**

1. Create a MS VisualStudio solution named **Assignment14**.
2. Create projects according to the assigned problems.
3. Use techniques and experience that you have obtained from previous chapters and this chapter.
4. Code and test your programs. They **MUST** be completed without syntax, logic and run-time errors.
5. Include comments of your name, date, and brief descriptions in all source codes.
6. Compress **Assignment14**folder into **ONE** zipped file.
7. Submit or re-submit your zipped file before its due date&time.

**View Rubric**

| Assignment(1) | | |
| --- | --- | --- |
|  |  |  |
| **Assignment(1)** | | |
| **Criteria** | **Ratings** | **Pts** |
| Comments with your name and date, and description |  | / 20 pts |
| Completion of requirements |  | / 20 pts |
| Syntax errors |  | / 20 pts |
| Logic errors |  | / 20 pts |
| Run-time errors |  | / 20 pts |
| Total Points: 0 | | |

int main()  
  
{  
    do  
    {  
        switch (menuOption())  
        {  
        case 0: exit(1); break;  
        case 7: Challenge7(); break;   
        case 8: Challenge8(); break;   
        case 9: Challenge9(); break;  
        case 11: Challenge11(); break;  
        case 12: Challenge12(); break;   
        default: cout << "\t\tERROR - Invalid option. Please re-enter."; break;  
        }  
        cout << "\n";  
        system("pause");  
  
    } while (true);  
  
    return EXIT\_SUCCESS;  
}

7. PersonData and CustomerData Classes

Design a class named PersonData with the following member variables:

lastName

firstName

address

city

state

zip

phone

Write the appropriate accessor and mutator functions for these member variables.

Next, design a class named CustomerData, which is derived from the PersonData class. The CustomerData class should have the following member variables:

customerNumber

mailingList

The customerNumber variable will be used to hold a unique integer for each customer. The mailingList variable should be a bool. It will be set to true if the customer wishes to be on a mailing list, or false if the customer does not wish to be on a mailing list. Write appropriate accessor and mutator functions for these member variables. Demonstrate an object of the CustomerData class in a simple program.

1. PreferredCustomer Class

A retail store has a preferred customer plan where customers may earn discounts on all their purchases. The amount of a customer’s discount is determined by the amount of the customer’s cumulative purchases in the store.

When a preferred customer spends $500, he or she gets a 5 percent discount on all future purchases.

When a preferred customer spends $1,000, he or she gets a 6 percent discount on all future purchases.

When a preferred customer spends $1,500, he or she gets a 7 percent discount on all future purchases.

When a preferred customer spends $2,000 or more, he or she gets a 10 percent discount on all future purchases.

Design a class named PreferredCustomer, which is derived from the CustomerData class you created in Programming Challenge 7. The PreferredCustomer class should have the following member variables:

purchasesAmount (a double)

discountLevel (a double)

The purchasesAmount variable holds the total of a customer’s purchases to date. The discountLevel variable should be set to the correct discount percentage, according to the store’s preferred customer plan. Write appropriate member functions for this class and demonstrate it in a simple program.

Input Validation: Do not accept negative values for any sales figures.

1. File Filter

A file filter reads an input file, transforms it in some way, and writes the results to an output file. Write an abstract file filter class that defines a pure virtual function for transforming a character. Create one derived class of your file filter class that performs encryption, another that transforms a file to all uppercase, and another that creates an unchanged copy of the original file. The class should have the following member function:

void doFilter(ifstream &in, ofstream &out)

This function should be called to perform the actual filtering. The member function for transforming a single character should have the prototype:

char transform(char ch)

The encryption class should have a constructor that takes an integer as an argument and uses it as the encryption key.

11. Course Grades

In a course, a teacher gives the following tests and assignments:

A lab activity that is observed by the teacher and assigned a numeric score.

A pass/fail exam that has ten questions. The minimum passing score is 70.

An essay that is assigned a numeric score.

A final exam that has 50 questions.

Write a class named CourseGrades. The class should have a member named grades that is an array of GradedActivity pointers. The grades array should have four elements, one for each of the assignments previously described. The class should have the following member functions:

setLab: This function should accept the address of a GradedActivity object as its argument. This object should already hold the student’s score for the lab activity. Element 0 of the grades array should reference this object.

setPassFailExam: This function should accept the address of a PassFailExam object as its argument. This object should already hold the student’s score for the pass/fail exam. Element 1 of the grades array should reference this object.

setEssay: This function should accept the address of an Essay object as its argument. (See Programming Challenge 6 for the Essay class. If you have not completed Programming Challenge 6, use a GradedActivity object instead.) This object should already hold the student’s score for the essay. Element 2 of the grades array should reference this object.

setPassFailExam: This function should accept the address of a FinalExam object as its argument. This object should already hold the student’s score for the final exam. Element 3 of the grades array should reference this object.

print: This function should display the numeric scores and grades for each element in the grades array.

Demonstrate the class in a program.

12 . Ship, CruiseShip, and CargoShip Classes

Design a Ship class that has the following members:

A member variable for the name of the ship (a string)

A member variable for the year that the ship was built (a string)

A constructor and appropriate accessors and mutators

A virtual print function that displays the ship’s name and the year it was built.

Design a CruiseShip class that is derived from the Ship class. The CruiseShip class should have the following members:

A member variable for the maximum number of passengers (an int)

A constructor and appropriate accessors and mutators

A print function that overrides the print function in the base class. The CruiseShip class’s print function should display only the ship’s name and the maximum number of passengers.

Design a CargoShip class that is derived from the Ship class. The CargoShip class should have the following members:

A member variable for the cargo capacity in tonnage (an int)

A constructor and appropriate accessors and mutators

A print function that overrides the print function in the base class. The CargoShip class’s print function should display only the ship’s name and the ship’s cargo capacity.

Demonstrate the classes in a program that has an array of Ship pointers. The array elements should be initialized with the addresses of dynamically allocated Ship, CruiseShip, and CargoShip objects. (See Program 15-14, lines 17 through 22, for an example of how to do this.) The program should then step through the array, calling each object’s print function.