COP3014-Foundations of Computer Science
Assignment #10

Objectives:

1. **Implement a friend function of a class;**
2. **Overload the << operator as a friend function of a class with chaining;**
3. **Overload the – operator as a member function of a class with chaining;**

This assignment is an extension of Programming Assignment 9 (Module 10's Programming Assignment). **You will implement class called "order_class".** The class will manage a dynamic array of purchase order records**. Called the program for this assignment "nursery_orders10.cpp".** I have provided a skeleton (driver) of "nursery_orders10.cpp" to help you implement this program.

Your input data will be in the file "nursery_stock.txt". The descriptions of the functions you will implement are as follows:

1. **(You implemented this function in the previous program.) the default constructor** to initialize the state of your class. The default constructor will read the data from the file "nursery_stock.txt" into the dynamic array STR. If STR becomes full, the function should call the function "double_size" to double the size (capacity) of STR. Remember, count, size, and STR are private members of your class and do not need to be passed to a member function of your class.
2. **(You implemented this function in the previous program.) is_Empty** is a Boolean public member function of the class. It has no formal parameters because count is a member of the state of the class (private member) and does not need to be passed to it because the state of the class is known to all member functions of the class. If count == 0 then true is returned; otherwise false is returned.
3. **(You implemented this function in the previous program.) is_Fu**ll is a Boolean public member function of the class. It has no formal parameters because count and size are members of the state of the class (private members) and they do not need to be passed to it because the state of the class is known to all member functions of the class. If count== size then true is return; otherwise false. The size is the capacity which is the total number of cells allocated to STR.
4. **(You implemented this function in the previous program.) search** is an integer public member function that has only one formal parameter, the key. key is the plant name for the record you are searching for. The array of records, STR and count are members of the state of the class and do not need to be passed to a member function of the class; The function will return the location of key in STR if it is there; otherwise -1 is returned.
5. **(You implemented this function in the previous program.) add** is a void public member function that inserts information for an order record into STR. Duplicates plant names are okay; add will prompt the user for the plant name (pname), county name (cname), plant cost (plant_cost), and the quantity (quantity. You may call process record to re-process STR when you add a new record. add has no formal parameters. Remember, to call the function double_size if STR is full before you add a new order record.

6. **Overload the operator "-" as member function of order_class with chaining. This function will have the same functionality as the "remove" function from the previous assignment. Recall the following about the function remove: "remove** is a public member function that deletes all records with the plant name that matches the value stored in key. If duplicate records exist with the same plant name they must all be deleted. "remove" has only one formal parameter, the key." **Note, because we are overloading with chaining we must return the current object "*this".**
7. **(You implemented this function in the previous program.) double_size** is a void public member function that doubles the size (capacity) of STR. "double_size" has no formal parameters because size, count and STR are all members of the state of the class, order_class. First, size is multipled by two; second, memory is allocated using the statement "order_record *temp = new order_record[size]; third the records in STR are copied into temp with the statement "temp[i]=STR[i]" using a for loop. Forth, the old memory for STR is de-allocated using "delete [ ] STR"; Finally, STR is set to point to the new memory pointed to by temp using "STR = temp".
8. **(You implemented this function in the previous program.) process** has no formal parameters because they are already declared in the state of the class. The function will calculate the net cost of the purchase (net_cost), the taxrate (tax_rate), the tax on the purchase (purchase_tax), the discount on the purchase (discount), and the total cost (total_cost). Please consider the following information to help you implement the necessary calculations:

    a. The tax rate (in percent) on a purchase is based on the county where the purchase was made. If the county was dade, the tax rate is 6.5%; if the count is broward, the tax rate is 6%; if the county was palm, the tax rate is 7%.

    b. The net cost of an order (net_cost), which does not include tax, is calculated by the following:

        net_cost = (quantity x plant_cost)

    c. The discount is based on the quantity of plants in the purchase.
    The discount is determined as follows:
        - If quantity equals 0, then the discount percentage is 0% of the net cost;
        - If 1<=quantity<=5 then discount percentage of the net cost is 1%;
        - If 6<=quantity<=11 then discount percentage of the net cost is 3%;
        - If 12<=quantity<=20 then discount percentage of the net cost is 5%;
        - If 21<=quantity<=50 then discount percentage of the net cost is 8%;
        - If quantity >05 then discount percentage is 13%;

        Note: Apply discount percentage after the net cost has been calculate

        discount = net_cost * (discount_percentage) / 100;
        (drop / 100 if you converted the discount_percentage to a decimal)

    d. The tax on a purchase (order_tax) is calculated by the following formula:

        purchase_tax = (net_cost * tax_rate / 100 *(drop /100 if you converted the rate from a percentage)*

e. The total cost of a purchase (rounded to the nearest hundredth) is calculated by the following formula:

total_cost = net_cost + purchase_tax – discount.
Note: All tax and cost calculations should be rounded to the nearest hundredths.

9. **Overload operator "<<" as a friend function of order_class with chaining. This function will have the same functionality as the "print" function in the previous assignment, except it will print to the screen. Recall the following about the function print: "print** is a public member function that has no formal parameters because count and STR are members of the state of the class. The function will also have one static local integer variable called "run". The function will print the value of run every time print is executed. The value of "run" should be incremented after you print its value the file. The function will print every field of every order_record in STR to the screen.

10. **(You implemented this function in the previous program.) the destructor** to de-allocate all memory allocated to STR. This function has no formal parameters because STR is a member of the state of the class; the destructor will be called automatically by the compiler.

You may implement more member functions if you find it necessary. Please start the assignment ASAP, and ask questions to make sure you understand what you must do. It is always good to start with the skeleton program (**nursery_orders10.cpp**) I provided. Remember to follow all style rules and to include all necessary documentation (consistent, indentation, proper variable names, pre/post conditions, program header, function headers, and so forth.) .

Output Format for the Function "operator<<":

1. Use the following format information to print the variables:

Field	Format
Plant Name	**string**
County Name	**string**
Plant Cost	**XXXX.XX**
Quantity of Plants	**XXXX**
Net Cost of Purchase	**XXXXX.XX**
Tax Rate	**X.XXX**
Purchase Tax	**XXXXX.XX**
Discount on Purchase	**XXXX.XX**
Total Cost of Purchase	**XXXXXXX.XX**

2. Consider the following sample output table when designing and implementing the function "**operator<<**":

(**The output is in the following order: plant name, county name, plant cost, quantity, net cost, tax rate, purchase tax, discount, total cost**).

owl	dade	10.55	100	1055.00	0.065	68.58	126.60	996.98
hibiscus	broward	15.82	15	237.30	0.06	14.24	11.87	239.67
rose	dade	9.99	45	449.55	0.065	29.22	35.96	442.81
carnation	palm	7.99	32	255.68	0.07	17.90	20.45	253.12

3. **Input Stream**
   In the assignment you will declare one ifstream to bind your input to the file "***nursery_stock.txt***" to an input file stream. Whenever a program performs file i/o you must include the "***fstream***" library. **Add the following statements to your program:**

   For source file, "***nursery_porders10.cpp***"

   - Add "#include <fstream>" to your #include statements in your source file.
   - Add "#include <string>" to your #include statement in your source file.
   - Add "#include <iomanip>" all formatting of output

4. Copy of Nursery_stock.txt:

owl dade 10.55 100
hibiscus broward 15.82 15
rose dade 9.99 45
carnation palm 7.99 32
rose palm 7.99 60
widow palm 25.75 5
carnation dade 12.55 10
carnation dade 12.55 8
lilly broward 6.92 150
xerabtgemum palm 13.63 50
yarrow dade 22.85 20
zenobia palm 37.19 32
zephyranthes broward 62.82 40
daisy broward 15.99 80
aconitum dade 30.02 72
amaryllis dade 16.14 65
bogonia broward 18.45 3
bellflow broward 2.96 200
bergenia palm 85.92 10