COP3014-Foundations of Computer Science
Module 10 Programming Assignment

Submit Assignment On Time!

Objectives:

1. **Separate class code into a declaration (header) and implementation components;**
2. **Implement a copy constructor;**
3. **Use the preprocessor directives #ifndef, #define and #endif;**

This assignment is an extension of Programming Assignment 10 (Module 11's Programming Assignment). **You will implement class called "order_class".** The class will manage a dynamic array of purchase order records. **Called the program (driver) for this assignment "amazon_porder11.cpp".** You must separate the declaration and implementation for the class "order_class". **Put the declaration in the file called "order_class.h", and put the class implementation in the file called "order_class.cpp".** However, you will need to modify the driver to test the functionality of your program. I have provided a driver skeleton to help you implement this program.

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

1. the copy constructor will perform a deep copy of an object.
2. **(you implemented this in the previous program) the default constructor to initialize the state of your class.** The default constructor will read the data from the file "purchase_data.txt" into the dynamic array INV. If INV becomes full, the function should call the function "double_size" to double the size (capacity) of INV. Remember, count and INV are private members of your class and do not need to be passed to a member function of your class.
3. **(you implemented this in the previous program) is_Empty** is a Boolean public member function of the class. It has no formal parameter 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.
4. **(you implemented this in the previous program) is_full** 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 INV.

5. **(you implemented this in the previous program) search** is an integer public member function that has only one formal parameter, the key. key is the cell phone number for the record you are searching for. The array of records, INV 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 INV if it is there; otherwise -1 is returned.
6. **(you implemented this in the previous program) add** is a void public member function that inserts information for a order record into INV. Duplicates cell numbers are ok; add will prompt the user for the cell number, item number, quantity, price, and processing plant. You may call process record to re-process INV when you add a new record. add has no formal parameters.
7. **overload operator "-" as a member function of order_class with chaining. This function will have the same functionality as the "remove" function. Recall the following about the function remove: "remove** is a void public member function that deletes all records with the cell number stored in key. If duplicate records exist with the same cell number they must all be deleted. "remove" had only one formal parameter, the key." **Note, because we are overloading with chaining we must return the current object "*this".**
8. **(you implemented this in the previous program) double_size is a void public member function that** doubles the capacity of INV. "double_size" has no formal parameters because size, count and INV 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 call_record[size]; third the records in INV are copied into temp with the statement "temp[i]=INV[i]" using a for loop. Forth, the old memory for INV is de-allocated using "delete [ ] INV"; Finally, INV is set to point to the new memory pointed to by temp using "INV = temp".
9. **(you implemented this in the previous program) process** has two input parameters: INV and count. The function will calculate the order tax rate (tax_rate), net cost of an order (net_cost), the tax on an order (order_tax) and the total cost of an order (total_cost) using the quantity, cost of an item (price), and processing plant identification number (processing_plant) for a purchase order record (order_record). Remember, we are using an array of records. Please consider the following:

   a) The tax rate on an order (tax_rate) is simply based on the processing plant identification number (processing_plant) which is where the order was processed (0<= processing_plant <= 50 then tax_rate = 6%; 51<= processing_plant <=110 then tax_rate = 7%; 111<= processing_plant<=200 then tax_rate = 8%; 201<=processing_plant<=500 then tax_rate = 9%; processing_plant >500 then tax_rate =11%) .

b) The tax on an order (order_tax) is calculated by the following formula:

order_tax = (quantity) x (price) x [(tax_rate) / 100] //Hint:
INV[i].order_tax = (INV[i].quantity * iNV[i].price ….

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

net_cost = (quantity) x price

d) The total cost of an order (rounded to the nearest houndredth) is calculated by the following formula:
Total_cost = net_cost + order_tax   All tax and cost calcuations should be rounded to the nearest hundredths.

10. **overload operator "<<" as a friend function of order_class with chaining. This function will have the same functionality as the "print" function (it will print INV to the screen). Recall the following about the function print:** **"print** is a void public member function that has no formal parameters because count and INV are members of the state of the class. The function will print every field of every order_record in INV to the screen".

11. **the destructor** to de-allocate all memory allocated to INV. This function has no formal parameters because INV is a member of the state of the class; it will be called automatically by the compiler.

Use the driver "amazon_porders11.cpp" to help you implement this program.

Output Format for the overloaded "operator<<":

Consider the sample output table below when designing and implementing the overloaded operator "operator<<". The output should be in the following order: cell phone number, item, quantity, price, processing plant, tax rate, order tax, net cost, total order cost). Use the information in the data file "purchase_data.txt" to produce the output. Your output should not contain any titles, but the output must be in the proper order.

```
9546321555    452-XLY      2   70.82   503    11    15.58   141.64   157.22
9546321555    742-6Z3    300   10.14    47     6   182.52  3042.00  3224.52
5612971340    924-YUT      8   23.00    51     7    12.88   184.00   196.88
```

3051234567	9L3-11T	47	50.12	200	8	188.45	2355.64	2544.09

Input Stream:

In the assignment you will declare one ifstream to bind your input to the file "purchase_data.txt" Whenever a program performs file i/o you must include the "fstream" library. Add the following statements to your program:

For source file, *"amazon_porders11.cpp"*:

- Add "**#include <fstream>**" to your # include statements in your source file.
- Add **"include <string>** to your # include statements in your source file.

Format of the input data file(input filename is "*purchase_data.txt*"): **Do not include column titles**

(cell phone number, item, quantity, price, processing plant)

9546321555	452-XLY	2	70.82	503
9546321555	742-6Z3	300	10.14	47
5612971340	924-YUT	8	23.00	51
3051234567	9L3-11T	47	50.12	200
7542346622	453-TTT	92	25.25	110
3054432762	213-ABC	15	105.02	111
9544321011	321-XMZ	112	32.41	92
8776219988	444-SSS	100	12.12	86
9042224556	290-P23	297	9.62	500
7877176590	23Z-00Q	32	9.62	201
5617278899	893-42T	642	0.81	72
9546321555	008-LL5	90	45.80	18
5612971340	452-XLY	42	99.99	503
3051234567	742-6Z3	14	10.14	47
7542346622	742-6Z4	13	66.48	3
3054432762	742-6Z5	209	0.77	2
9544321011	742-6Z6	108	1.17	1
8776219988	213-ABC	67	105.02	111
9042224556	321-XMZ	500	32.41	92
7877176590	444-SSS	43	12.12	86
5617278899	290-P23	16	9.92	500
8776219988	23Z-00Q	82	9.62	201
9042224556	23Z-00Q	87	9.62	201
7877176590	23Z-00Q	445	9.62	201

Handing in your program

Electronically submit ***the source file "amazon_porders11.cpp" and the datafile "purchase_data.txt" to*** Canvas before the due date and time. **Remember, to submit an assignment – even if it is late…..**