# Programming Project 7
ECE312 Spring 2023
CRM – part 2 in C++

**Objectives:** In this project, you'll use your newly acquired C++ programming skills to create an improved version of the CRM project. This will involve writing two C++ classes: the Customer and the CustomerDB. We will also use a provided String class.

The project will involve essentially the same data structures and algorithms from before, but hopefully, this time around you'll discover that (a) the program isn't nearly as tedious (no more destroyString!) and (b) the code you produce is a whole lot shorter and easier on the eyes. Good luck!

**Requirements:** You will do the following for this project: edit CustomerDB.cpp to complete all member functions for the class, and edit Project7.cpp to implement processInventory, processPurchase, and processSummarize.

First, you should read the provided files. You will note that both files to be edited already have some code. While you're not explicitly required to use the provided code, your solution may be a good deal shorter, cleaner and simpler when using it.

For CustomerDB.cpp, you must write two functions. The first of these functions CustomerDB::isMember searches through the current set of Customers and returns true if it finds a Customer with the matching name, and returns false otherwise. The second function, CustomerDB::operator[](String) must do the following:

- If a Customer in the CustomerDB has a name that matches the argument to operator[], then your function must return that Customer (returning by reference).
- If there is no Customer in the CustomerDB with that name, then your function must ADD a new Customer to the database and then return a reference to that newly added Customer.
- If you add a new Customer to the database, you must ensure that there is capacity for the customer in the array, and you must use amortized doubling to resize the array if there is insufficient capacity.

This is separate from the operator[] which takes an int, which simply returns the customer at a given position in the array. This adds a new method of accessing the CustomerDB, like: database["Phil"].bottles += 5;.

For Project7.cpp, you must write an implementation for processPurchase, processSummarize, and processInventory. Part of your grade will be determined by how well these functions utilize the CustomerDB. As a general hint, if you can make those functions shorter, you're (probably) making them better.

You **MUST NOT** change CustomerDB.h, UTString.h or Customer.h! If you do, your submission will likely fail.

**Submission:** You **MUST** submit both CustomerDB.cpp and Project7.cpp. Zip up the files and rename the zip file Project7_<eid>.zip, where <eid> is replaced with your own UT EID. Capitalization here doesn't matter.

**Testing:** The same three test files from last time are provided. You might want to add some additional tests of your own. You are, of course, responsible for ensuring that your program is correct (i.e., not just able to execute the three test inputs and produce the correct output through some coincidence).

**CHECKLIST – Did you remember to:**

- ☐ Re-read the requirements after you finished your program to ensure that you meet all of them?
- ☐ Make sure that your program passes all our testcases?
- ☐ Make up your own testcases?
- ☐ Upload your solution to Canvas?
- ☐ Download your uploaded solution into a fresh directory and re-run all testcases?

**Answers to FAQ**

1. No customers will attempt to buy negative numbers of anything.
2. Don't add customers who ask for 0 items of anything. Don't respond to them either.
3. If a customer asks for more than the available inventory, don't add them if they are not already a customer. You must however, respond to them with "Sorry …".
4. Use the plural forms of Diapers, Rattles, and Bottles even if it should be singular. Capitalization matters, as well.
5. Only use cout to print. Use the c_str() function of the string class to get a string that cout can print properly (ex: std::cout << foo.c_str();).
6. Amortized doubling: Double the array whenever it is full so that you can add more elements next time. You'll keep track of array capacity and length separately so that you know when the array is full.
7. Turn in everything you edit. Zip up all the files and rename the zip file Project7_<eid>.zip, where <eid> is replaced with your own UT EID.