Program Purpose

Practice designing, implementing program solutions in C++. Debugging, file IO, data structures, UNIX.

Mandatory Instructions

Your assignment is to create the solution for the following **interactive** program. Please read the program description carefully and use the suggestions below. Although your program will receive input from the keyboard, your final photo on BGLinux will get input from a file through a special command that will simulate user input. No **fstream** include should be present in this program!!!

Name the program file: **prog2.cpp**

Program Design

1. Define a struct called Customer that contains members as outlined below:

```
struct Customer
{
   int    id;      // id
   string name;    // Name
   string zip;     // ZIP code
   double balance; // Account balance
   string pmtDate; // Date of last payment
};
```

2. Your *main()* function should do the following:

   a. Declare an array of 50 structures to hold customer data.
   b. Call function **menu** to get the valid menu choice from the user. The function has no parameters but will return a number between 1 and 5. USE DATA VALIDATION in the function to make sure the value provided by the user is valid. The menu function should display the following menu:

```
              Menu of choices
++++++++++++++++++++++++++++++++++++
1. Enter new account information
2. Change account information
3. Display all account information
4. Find customer
5. Exit the program
```

   c. In main create a loop that will execute until the user enters **5** to exit the program. Inside the loop your code should do the following:

      i. Use an if statement to determine which choice should be executed (1-4):

         1. If the choice is 1 invoke a function called **addCust** to add a new customer to the end of the list. Check to make sure there is room in the array **before** calling this function. Use an **if** statement. Use a global constant called **MAXCUST** in the condition. This function's arguments will be the array, and the number of current customers.

         2. If the choice is 2, invoke function **updateCust** to update a customer's data. This function's arguments will be the array, and the number of current customers.

         3. If the choice is 3, invoke function **displayCust** to print all of the information in tabular form. Decide on what arguments need to be passed to this function.

4. If the choice is 4, invoke function **findCust** to locate a customer in the array of customers. This function should not output anything to the console window. Main, based on the return value from this function, should show all customer's details.

Your solution must contain the following functions:

d. Function **addCust**:
This function should ask the user to enter data for a customer and add it to the bottom of the array. This function **must** not accept negative values for the balance. Use a data validation loop. Do not allow customer id to be reused, i.e., cannot be shared between two customers. User id cannot be negative.

e. Function **updateCust**:
This function should ask the user for the customer's ID. Function **findCust** should be called to locate the customer (see description below). The return value from **findCust** should be used to update the customer's information or state that the customer was not found if appropriate. To update the data, the **updateCust** function should ask the user to enter new data for the customer. Do not accept negative values for the balance. Use a data validation loop. User Id should not be updatable so don't ask for the id to be entered by the user.

f. Function **displayCust**:
This function will print all customer data in tabular form. Utilize some headers above each column.

g. Function **findCust:**:
This function will accept the array, the number of customers and the id of the customer to be found. The function will return the subscript or -1 if not found. The function should not display any output. This function should utilize binary search.

h. Function **sortCust:**
This function should implement a selection sort by id value. It should be called before any search of customers is performed. This function should not display any output.

Program Documentation

Provide documentation header for your program and for each function other than main() as discussed in class.

What to turn in?

IMPORTANT. You will not change your code to read from a file, **keep the cin statement**, instead, you will need to **echo** your cin statement before you make the photo. To do this you will place a cout statement after every cin statement. This will echo all of the data entered. For example:

```
cout <<  "Enter your choice: ";
cin >> choice;
cout << choice << endl;

cout <<  "Enter a customer name: ";
getline(cin, name);
cout << name << endl;
```

***Note: This will allow the TA to see the input that will be redirected from a file using a special BGLinux command, so you do not have to type all of the data in yourself.**

To receive full credit for this assignment you need to create the following photo on the due date:

1. **A photo with the following**.
Before you create the photo, copy the **prog2.txt** from the lib directory on BGLinux to your cs2020 directory.

a. $ photo prog2.log
b. $ ls -o
c. $ rm ./a.out
d. $ cat prog2.cpp
e. $ g++ prog2.cpp
f. $ ./a.out **< prog2.txt** // this replaces entering a bunch of user input at command prompt
g. **$**ctrl-d
h. $ exit