**3307 Assignment 1**

**Group 1**: Valmir Verbani   
**&&** Zaid Albirawi

1. **Deliverable 1:**
   1. Client:
      1. Create/Log in to a client account.
      2. Create a chequing/saving account (initially 0 balance).
      3. Deposit an amount into an account.
      4. Withdraw an amount such that it doesn’t exceed balance amount total.
      5. If the chequing accounts balance falls under 1000$ in a withdraw operation, then the user is charged 2$ and the operation is only finalized upon the user’s agreement to that fee.
      6. Transfer a balance amount from one account to another.
      7. Close an account saving/chequing, if and only if the balance is 0.
   2. Manager:
      1. Open a client’s account.
      2. Close client’s account, if and only if the balance is 0.
      3. Manage accounts, preform any operation a client is capable of performing.
      4. View a specific client’s information.
      5. View multiple clients’ information.
   3. Maintenance:
      1. Toggle the execution trace ON/OFF.
      2. Print execution trace
2. **Deliverable 2:**

Our project consists of a “save and load function”, two requirements that are beyond the core of the project description. These two functions were created because for any program that takes in a distinguishable ‘id’, especially with a password, should be able to recognize this client even after the program is closed then reopened. The process is simple, the load function is called at the beginning of the main function which loads the client’s account information and the previous trace output, from a text file. At the end of the main function, the save function is called to save the client’s account information and the current trace output, to a text file.

1. **Deliverable 3:**

In order to run this program, you need to be open the command prompt in the correct directory where the code related files are located. Once in the proper directory, compile the program by typing in the following command, “g++ 3307Assignment1.cpp”. Once the program is compiled, type in “./a.out” to start the program. Once the program starts, simply follow instructions. If you want to log in as a manager or a maintenance person, the usernames and passwords are:

For manager log-in:  
 **Username**: manager  
 **Password**: password

For Maintenance log-in:  
 **Username**: maintainer  
 **Password**: password

There are no default client user, therefore, you would have to create one using the manager account.

1. **Deliverable 4:**

Five Scenarios: Logging in with manager id and creating a client account , creating an saving account for a client, depositing an balance onto the newly created savings account, transferring a certain amount to a different account that is not already created, and trying to close an account with a balance of greater than 0$.

Five Scenarios:

1. A manager creating/opening an account for a client. For a client to have a saved username and password the user must first log in as a manager with the username ‘manager’ and the password ‘password’. In the first scenario we create a username ‘val’ with the password ‘ password’. See Diagram A.
2. In the second scenario we see that we have created a client account but we have not assigned/ created savings or chequing account for that client. Here we will create the client ‘val’ a savings account. Once we log in as the manager, we will click 3 (Manage account) from the 6 options. Then it will ask you which clients account you want to manage. Here we will type in 1. Since we do not have a savings account, it will ask us if we want to create a savings account, which we will type in ‘y’ in order to create it. See Diagram B.
3. In the third scenario we have a savings account with a 0$ balance. It is easy for the manager to deposit a specified amount to the clients account. Once you are logged in as a manager, choose 3 (Manage account) from the 6 options. Then it will ask you which clients account you want to manage, in our class we will type in ‘val’. Then click 1 for saving account. Then it will ask you to pick from another 6 options, for deposit we will pick option 2. After we picked this option, we will type in a dollar amount value to deposit into our account. In this case the deposit amount will be 1500$. See Diagram C.
4. In the fourth scenario, we are logged in as a manager, we have a savings account with no chequing account. Our savings account has 1500$ but we want to transfer 1000$ onto our chequing. Once we are logged as a manager and we typed in the clients username we want to manage. It will ask us what we want to manage, in this case we will pick 4 (transfer), then we will type in ‘y’ for yes. After typing ‘y’, we will type in the amount we want to transfer to the chequing account, which is 1000$. See diagram D.
5. In the last scenario, we are also logged in as a manager with both savings and a chequing account with balances on both accounts. In this case, out the 6 options to pick from, we will pick number 5 (close account). In this case it will not let us close the account because the two balances both have a balance that is greater than 0$. See diagram E.

Diagram A)

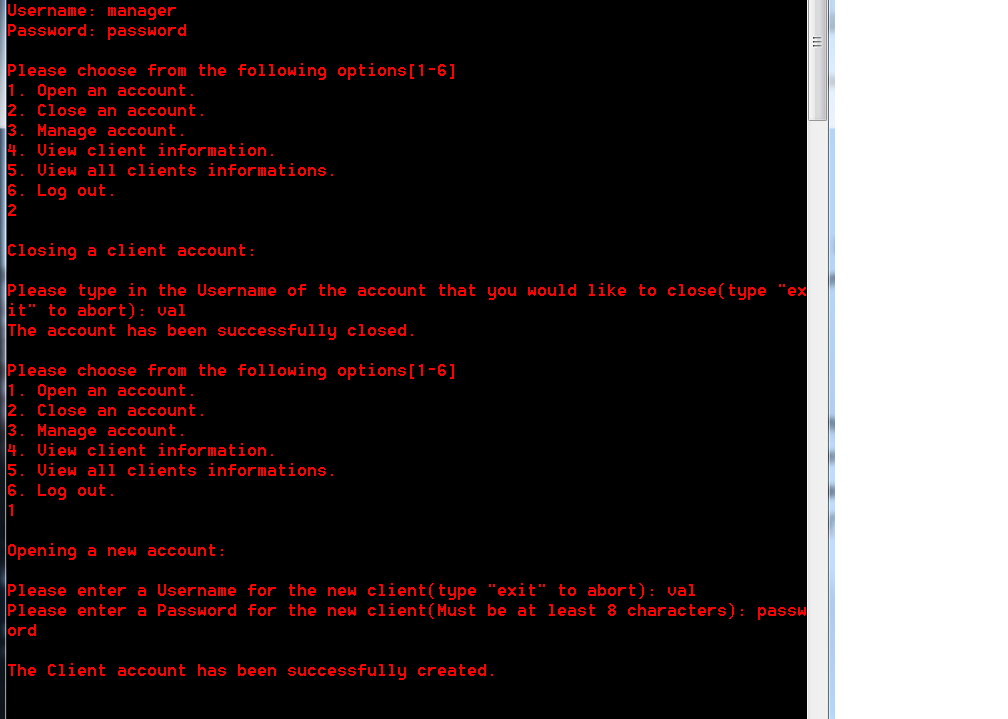


Diagram b)

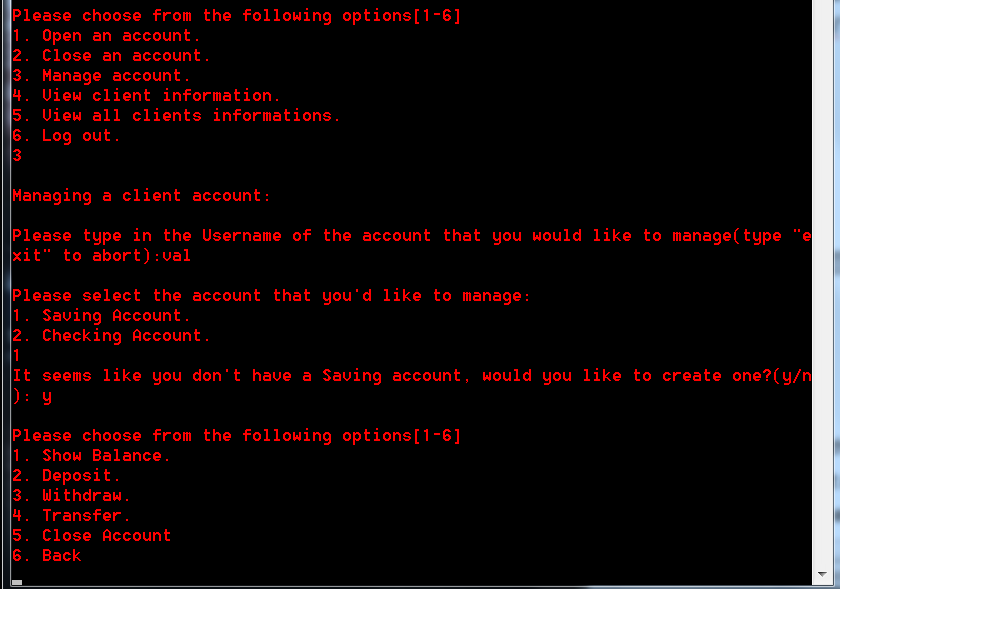


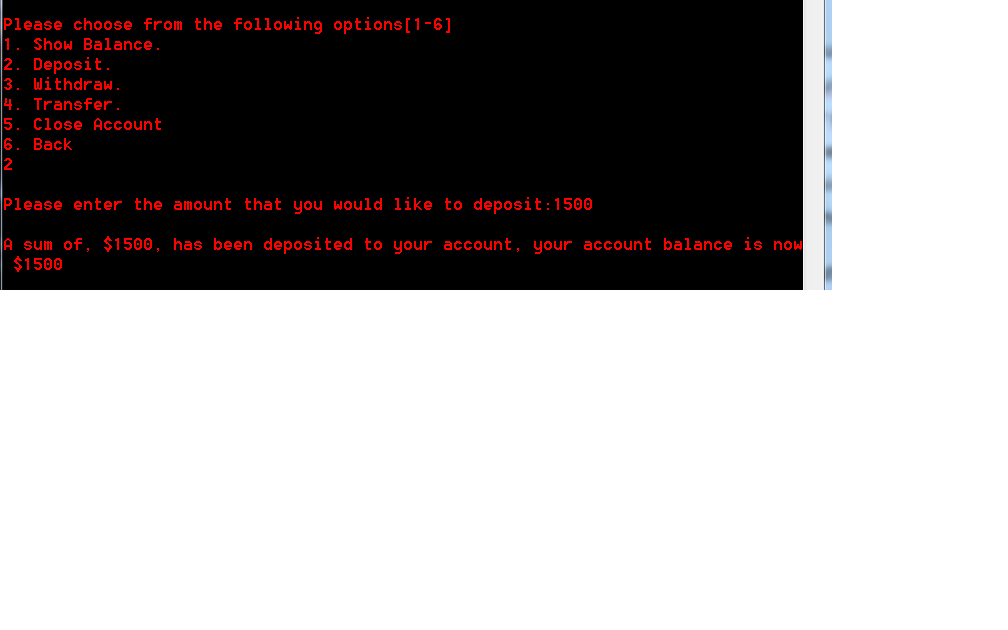
Diagram C)

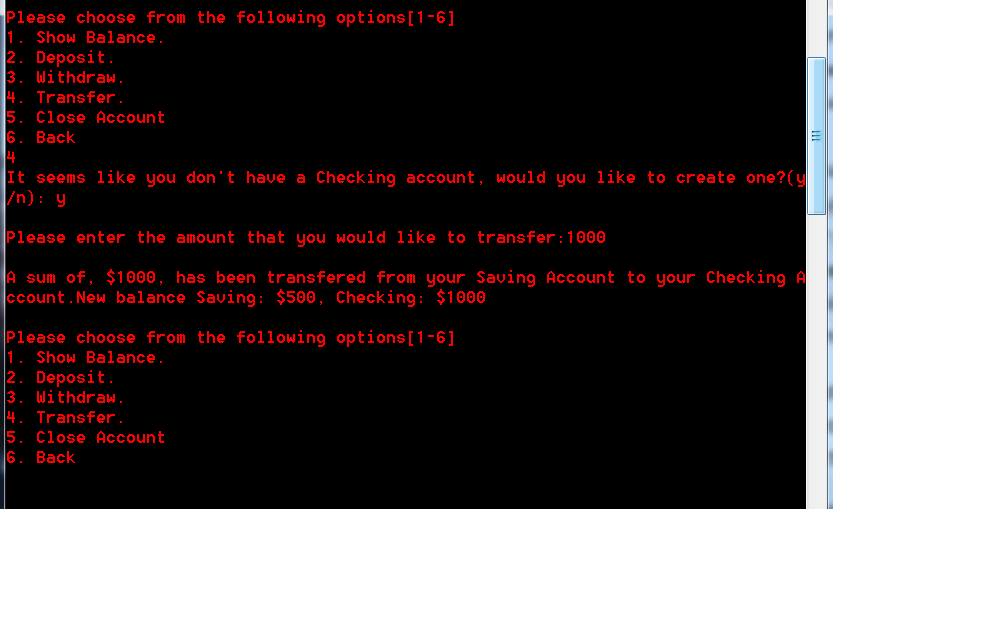
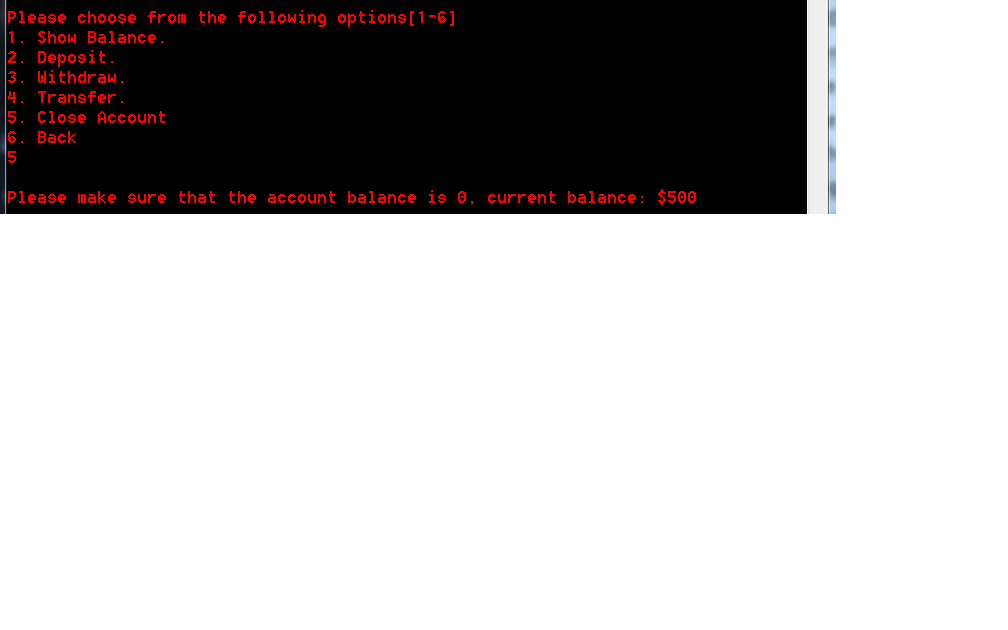
Diagram D)

Diagram E)

1. **Deliverable 5:**

Valmir:

I have learned that c++ has a lot more similarities to java than I had initially expected. With some research I have also learned how to properly save data to an external text file format document and then loading the saved data properly and pushing onto a vector so that the data isn’t lost or corrupted.

Zaid:

I have learned a lot about the differences between Java and C++, mainly, the way in which objects are created, the way inheritance works in C++, and the fact the pass by value on an object in C++ is really pass by value, where pass by value in Java is really pass by reference. Furthermore, C++ gives the developer more freedom with the OO design choices in comparison with Java. What we could’ve done differently is mostly change the way we handle different operations. Instead of dividing the operations by users, we could build all the operations separately and then pass them to each user depending on the user’s account type.