Discipline of Computing and IT
University of Newcastle

SENG1120/6120 – Semester 1, 2018
Lab 2 (Week 2)

Video guide: https://www.youtube.com/watch?v=vHHSGAhl1mY

1. Download the provided files (`account.h`, `account.cpp` and `bank.cpp`) on Blackboard to the folder in which you will perform this week's laboratory exercises.

2. As shown in lectures, modify the code to include a macro guard for the class account.

3. Ensure that your program compiles and runs as it should. Modify the `makefile` provided and put the two `.cpp` files next to "`SOURCES`".

4. Now, in the file `bank.cpp`, introduce the variable `ptr` that is a pointer to the account object `my_account1`. Change all calls to `my_account1` so that they are achieved using `ptr->`.

6. Just for interest, use a `cout` statement to display `ptr, &ptr` and the balance in the account pointed to by `ptr`.

7. Use the = operator to create a copy, called `my_copy`, of `my_account1`. Check that `my_copy` is really a copy, by calling its mutator methods and confirming that the calls do not alter `my_account1`.

8. Now create a reference variable called `acc2` that implements an alias for `my_account1`. Verify that `acct2` is a reference type by demonstrating that calls on the mutator methods of `acct2` also mutate the contents of `my_account1`.

9. Include a function in `bank.cpp` called `compareAccRef` that takes, as parameters, references to two instances of account, and returns a reference to the instance that contains the bigger balance. Verify that the returned value is a reference by using it to query the instance's balance.

10. Now re-do step 9, this time using pointers. Create a method `compareAccPtr` that takes two pointers and returns a pointer to the instance with the bigger balance. Use the returned pointer to query the instance's balance.

FOR SENG6120 (or if you want to learn more)

11. Create a new class called `portfolio` that has, as member variables, two `account` objects representing, respectively, savings and cheque accounts. After defining and implementing the constructor for portfolio, define and implement mutator methods `savings_zero, cheque_zero, savings_deposit, cheque_deposit, savings_withdraw` and `cheque_withdraw`. Then

define and implement the query methods `savings_bal` and `cheque_bal`. Ensure that `portfolio` is protected by a guard compiler directive.

12. Write a program that creates an instance of `portfolio`, and then uses its methods to demonstrate the success of your implementation.

Good Luck!