Lab: Inheritance

Implement a base class Account and derived classes Savings and Checking. In the base class, supply member functions deposit and withdraw. Provide a function dailyInterest that computes and adds the daily interest. For calculations, assume that every month has 30 days. Checking accounts yield interest of 3 percent monthly on balances over $1,000. Savings accounts yield interest of 6 percent on the entire balance. Use the supplied driver program that makes a month's worth of deposits and withdrawals and calculates the interest every day.

```
Use the code below and modified the code to implement the requirement
of the lab


Define constant for DAYS_PER_MONTH = 30;
Define constant SAVING_RATE = 0.06;


class Account
{
public:
   Account();  // doing nothing
   Account(double b); // set balance to b
   void deposit(double amount); // balance is increased by amount
   void withdraw(double amount); // balance is reduced by amount
                                 // make sure you can't withdraw if
                                 // amount is larger than balance
   double getBalance() const; // return the account balance
private:
   double balance;
};
```

You need to implement all the member functions for the class Account

```
//...................................................................

// define class Savings which will inherit class
// object Account - modify ????? below with correct code

class Savings : ???? // modify ??? with correct code to inherent Account
{
public:
  Savings(); // doing nothing
  Savings(double b); // initialize balance to b
  void dailyInterest();
};
```

You need to implement all the member functions for the class Savings

```
/**
   Determine the daily interest and deposit it into the account.
*/
void Savings::dailyInterest()
{
   //
   // calculate the daily interest rate which is
   //   balance * SAVING_RATE / DAYS_PER_MONTH
   // Then call member function to deposit the interest to the
   // balance
   //
   // your code here to deposit the daily interest

}

//.................................................................

// define class Checking which will inheritance class
// object Account - modify ????? below with correct code

class Checking: ???? // modify ?????? with correct code to inherent Account
class Account
{
public:
  Checking();  // doing nothing
  Checking(double b);  // set balance to b
  void dailyInterest();
};
```

You need to implement all the member functions for the class Savings

```
/**
   Determine the daily interest and deposit it into the account.
```

Checking accounts yield interest of 3 percent monthly on balances over $1,000.

Calling dailyInterest() will calculate daily interest and add the daily interest to the balance

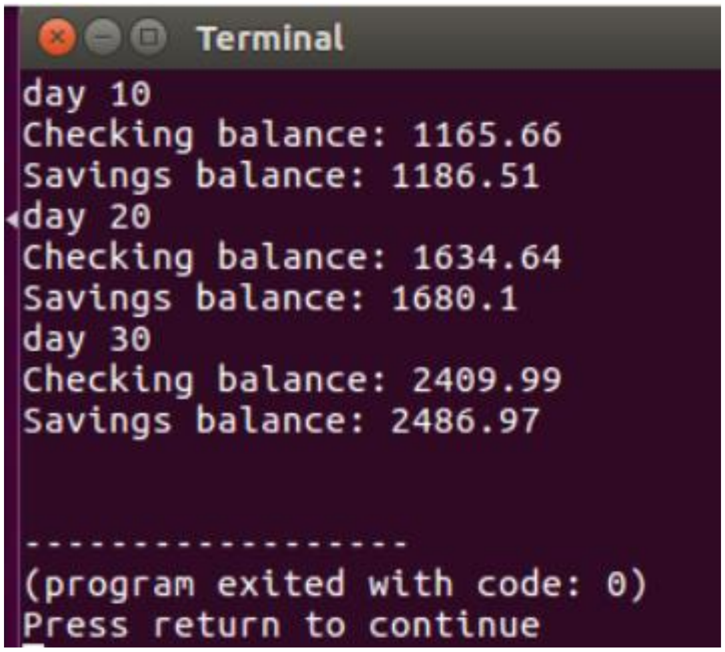
```
*/
void Checking::dailyInterest()
{
   const double RATE = 0.03;
   const double MIN_BALANCE = 1000;

/* your code here
   Calculate daily interest for any balance over $1000 and
   deposit the daily interest to the balance
*/

}
```

```
/*
 Use the following test template to test your program
*/
int main()
{
   Checking c = Checking(1000.0);
   Savings s = Savings(1000.0);
   for (int i = 1; i <= DAYS_PER_MONTH; i++)
   {
      c.deposit(i * 5);
      c.withdraw(i * 2);
      s.deposit(i * 5);
      s.withdraw(i * 2);
      c.dailyInterest();
      s.dailyInterest();
      if (i % 10 == 0)  // use % to only print out days 10, 20, 30
      {
         cout << "day " << i << "\n";
         cout << "Checking balance: " << c.getBalance() << "\n";
         cout << "Savings balance: " << s.getBalance() << "\n";
      }
   }
   return 0;
}
```

Your program should have similar output:



```
day 10
Checking balance: 1165.66
Savings balance: 1186.51
day 20
Checking balance: 1634.64
Savings balance: 1680.1
day 30
Checking balance: 2409.99
Savings balance: 2486.97


------------------
(program exited with code: 0)
Press return to continue
```

Your program submission:
1. Single pdf file contains all of the following:
a. Program description (the purpose of this program)
b. Include the source codes (must call PrintMeFirst() function)
c. Include the screen shots of the program output when you test the program. Your test cases must include the in the driver test program.

2. Include zip file contains all the files you used for this program a. Your source code must be properly documented with the following information:
• Description:
• Parameters
• Inputs/Outputs
• You can use block comments for section of codes and/or line comments