Details for Assignment 1 test outputs

1. Unit Test (2 points)

Tests that BankAccount("Jane",100.00, 500.00) correctly initializes bank account. Tests bankAccount.GetName().equals("Jane"), bankAccount.GetChecking() == 100.0, and bankAccount.GetSavings() == 500.0.

In [ ]:
```
bool testPassed(std::ofstream& testFeedback) {
   BankAccount bankAccount("Jane",100.00, 500.00);

   if (bankAccount.GetName().compare("Jane") == 0){
      cout << "GetName() correctly returned " << bankAccount.GetName() << endl;
   }
   else {
      cout << "GetName() incorrectly returned " << bankAccount.GetName() << endl;
      return false;
   }

   if (bankAccount.GetChecking() == 100.0) {
      cout << "GetChecking() correctly returned " << bankAccount.GetChecking() << endl;
   }
   else {
      cout << "GetChecking() incorrectly returned " << bankAccount.GetChecking() << endl;
      return false;
   }

   if (bankAccount.GetSavings() == 500.0) {
      cout << "GetSavings() correctly returned " << bankAccount.GetSavings() << endl;
   }
   else {
      cout << "GetSavings() incorrectly returned " << bankAccount.GetSavings() << endl;
      return false;
   }

   return true;
}
```

2. Unit Test (2 points)

Tests that BankAccount("Jane", 500.00, 1000.00) correctly initializes bank account.

Tests bankAccount.DepositSavings(123.00), bankAccount.GetSavings() == 1123.00, bankAccount.WithdrawSavings(25.00), and bankAccount.GetSavings() == 1098.00.

In [ ]:
```
bool testPassed(std::ofstream& testFeedback) {
   BankAccount bankAccount("Jane", 500.00, 1000.00);
   double amt1 = 123.00;
   double amt2 = 25.00;

   // check deposit to savings
   double savings = bankAccount.GetSavings();
   bankAccount.DepositSavings(amt1);

   if (bankAccount.GetSavings() == (savings + amt1)) {
      cout << "DepositSavings(123.00) passed: new balance is " << (savings + amt1) << endl;
   }
   else {
      cout << "DepositSavings(123.00) failed: new balance should be "
        << (savings + amt1) << ", not " << bankAccount.GetSavings() << endl;
      return false;
   }
   // check withdrawal from savings
   savings = bankAccount.GetSavings();
   bankAccount.WithdrawSavings(amt2);

   if (bankAccount.GetSavings() == (savings - amt2)) {
      cout << "WithdrawSavings(25.00) passed: new balance is " << (savings - amt2) << endl;
   }
   else {
      cout << "WithdrawSavings(25.00) failed: new balance should be "
        << (savings - amt2) << ", not " << bankAccount.GetSavings() << endl;
      return false;
   }

   return true;
}
```

3. Unit Test (2 points)

Tests that BankAccount("Jane", 750.00, 450.00) correctly initializes bank account.

Tests bankAccount.DepositChecking(75.00), bankAccount.GetChecking() == 825.00, bankAccount.WithdrawChecking(45.00), and bankAccount.GetChecking() == 780.00.

In [ ]:
```
bool testPassed(std::ofstream& testFeedback) {
   BankAccount bankAccount("Jane", 750.00, 450.00);
   double amt1 = 75.00;
   double amt2 = 45.00;

   // check deposit to savings
   double checking = bankAccount.GetChecking();
   bankAccount.DepositChecking(amt1);

   if (bankAccount.GetChecking() == (checking + amt1)) {
      cout << "DepositChecking(75.00) passed: balance is " << (checking + amt1) << endl;
   }
   else {
      cout << "DepositChecking(75.00) failed: balance should be "
        << (checking + amt1) << ", not " << bankAccount.GetChecking() << endl;
      return false;
   }

   // check withdrawal from savings
   checking = bankAccount.GetChecking();
   bankAccount.WithdrawChecking(amt2);

   if (bankAccount.GetChecking() == (checking - amt2)) {
      cout << "WithdrawChecking(45.00) passed: balance is " << (checking - amt2) << endl;
   }
   else {
      cout << "WithdrawChecking(45.00) failed: balance should be "
        << (checking - amt2) << ", not " << bankAccount.GetChecking() << endl;
      return false;
   }

   return true;
}
```

4. Unit Test (2 points)

Tests that BankAccount("Maria", 100.00, 100.00) correctly initializes bank account. Tests bankAccount.DepositChecking(-5.00)/bankAccount.WithdrawChecking(-5.00), bankAccount.GetChecking() == 100.00, bankAccount.DepositSavings(-5.00)/bankAccount.WithdrawSavings(-5.00), and bankAccount.GetSavings() == 100.00

In [ ]:
```
#include <cmath>

bool testPassed(std::ofstream& testFeedback) {
   BankAccount bankAccount("Maria", 100.00, 100.00);
   bankAccount.DepositChecking(-5.00);
   double amt = bankAccount.GetChecking();

   // check deposit to savings
   if (fabs(amt-100) > 0.1) {
      cout << "DepositChecking(-5.00) failed: negative deposits should not change 100.00 checking balance" << endl;
      return false;
   }
   else {
      cout << "DepositChecking(-5.00) passed: negative deposits did not change 100.00 checking balance" << endl;
   }

   bankAccount.WithdrawChecking(-5.00);
   amt = bankAccount.GetChecking();

   // check deposit to savings
   if (fabs(amt-100) > 0.1) {
      cout << "WithdrawChecking(-5.00) failed: negative withdraw should not change 100.00 checking balance" << endl;
      return false;
   }
   else {
      cout << "WithdrawChecking(-5.00) passed: negative withdraw did not change 100.00 checking balance" << endl;
   }

   bankAccount.DepositSavings(-5.00);
   amt = bankAccount.GetSavings();

   // check deposit to savings
   if (fabs(amt-100) > 0.1) {
      cout << "DepositSaving(-5.00) failed: negative deposits should not change 100.00 savings balance" << endl;
      return false;
   }
   else {
      cout << "DepositSaving(-5.00) passed: negative deposits did not change 100.00 savings balance" << endl;
   }

   bankAccount.WithdrawSavings(-5.00);
   amt = bankAccount.GetSavings();

   // check deposit to savings
   if (fabs(amt-100) > 0.1) {
      cout << "WithdrawSavings(-5.00) failed: negative withdraw should not change 100.00 savings balance" << endl;
      return false;
   }
   else {
      cout << "WithdrawSavings(-5.00) passed: negative withdraw did not change 100.00 savings balance" << endl;
   }

   return true;
}
```

5. Unit Test (2 points)

Tests that BankAccount("James", 1000.00, 1000.00) correctly initializes bank account. Tests bankAccount.TransferToSavings(100.00), and bankAccount.GetSavings() == 1100.00, and bankAccount.GetChecking == 900.00

In [ ]:
```
bool testPassed(std::ofstream& testFeedback) {
   BankAccount bankAccount("James", 1000.00, 1000.00);
   double savings = bankAccount.GetSavings();
   double checking = bankAccount.GetChecking();
   bankAccount.TransferToSavings(100.00);

   // check transfer to savings
   if (bankAccount.GetSavings() != (savings + 100.00)) {
      cout << "TransferToSavings(100.00) failed: savings should be $1100.00" << endl;
      return false;
   }
   else {
      cout << "TransferToSavings(100.00) passed: savings is $1100.00" << endl;
   }

   // check transfer from checking
   if (bankAccount.GetChecking() != (checking - 100.00)) {
      cout << "TransferToSavings(100.00) failed: checking should be $900.00" << endl;
      return false;
   }
   else {
      cout << "TransferToSavings(100.00) passed: checking is $900.00" << endl;
   }

   return true;
}
```