Kelly Tran

405808641

CS31

Project 6 Report

Obstacles

One obstacle that I had while making this project was how to make .h files and not .hpp files, which was something I did in the previous homework. I also had a hard time figuring out how to look through the ID numbers while making BloodDonation.cpp and check each index if the number was between 0 and 9 while making sure that the ID was a six-digit number. I realized later that I could just treat the constraints like a regular number, and I put bounds to the ID number instead of running through each digit and making sure it was a number between 0 and 9. This was only possible because a TA had said the project does not test for leading 0’s. I also had a hard time figuring out how to make the boolean to see if the balance should be increased in VacationAccount.cpp. I had to look back into the notes from Project 5 to understand how to add all the clauses under one if-statement.

Test Case

**BloodDonation doner1( 123456, 30, 125);**

**BloodDonation doner2( 99999, 69, 109.50);**

**BloodDonation doner3( 1234567, 78, 420.08);**

**BloodDonation doner4( 129308, 17, 99.99);**

**VacationAccount account(123456);**

assert( doner1.getID( ) == 123456);

I used this to test if the code would run.

assert( doner3.getID( ) == 1234567);

I used this to test if the code would fail because the number is above the bounds.

assert( doner2.getID( ) == 99999);

I used this to test if the code would fail because the number is below the bounds.

assert( doner2.getID( ) == -1);

I used this to see if the code would run because the code would recognize that the number is outside of the bounds and return -1.

assert(doner1. getAge( ) == 30);

I used this to test if the code would run because the age is within bounds

assert(doner2.getAge( ) == 69);

I used this to test if the code would fail given the age is greater than the stated bounds.

assert(doner4.getAge( ) == 17);

I used this to test if the code would fail given the age is less than the stated bounds.

assert(doner2.getAge( ) == -1);

I used this to see if the code would run because the code would recognize that the number is outside of the bounds and return -1.

assert(doner2.getWeight( ) == 109.50 );

I used this to test if the code would run because the weight is within bounds.

assert(doner3.getWeight( ) == 420.08 );

I used this to see if the code would fail because the number is greater than the bounds.

assert(doner4.getWeight( ) == 99.99 );

I used this to see if the code would fail because the number is less than the bounds.

assert(doner3.getWeight( ) == -1 );

I used this to see if the code would run and recognize that the number is outside the bounds and return -1.

assert( account.getBalance( ) == 4 );

I used this to see if the code would fail because there should be nothing in the balance for vacation account yet.

assert( account.getBalance( ) == 0 );

I used this to see if the code would run and recognize that there’s nothing in the balance yet.

assert( account.getID( ) == 123456);

I used this to see if the code would run and if the vacation account matches the ID number.

assert( account.getID( ) == 99999);

I used this to see if the code would fail because the vacation account doesn’t match the ID number.

assert( account.addVacationToAccount( doner2 ) == true );

I checked this to see if the code would fail because the donor number does not match the vacation account ID.

assert( account.addVacationToAccount( doner1 ) == true );

I checked this to see if the code would run because the donor number does match the vacation account ID.

assert( account.getBalance( ) == 4 );

I checked this to see if the balance in the account added 4 to the balance because doner 1 passes all of the stipulations to add to the balance.