Unit Testing

Customer:

Test Funtion: void depositAmount()

Test 1: Entering a large value

```
Customer customer;
initialAmount = customer.getAmount();
int amountToDeposit = 100000;
customer.depositAmount();

if(customer.getAmount() != (initialAmount + amountToDeposit)){
    cout << "Test 1 Failed" << endl;
}
```

User Input: 100000

Initial Amount: 350000

Expected Output: 450000

Program Output: 450000

Test 2: Entering negative value. If a negative amount is entered, program must not deposit the amount

```
Customer customer;
initialAmount = customer.getAmount();
int amountToDeposit = -1000;
customer.depositAmount();

if(customer.getAmount() != initialAmount){
    cout << "Test 2 Failed" << endl;
}
```

User Input: -1000

Initial Amount: 3500

Expected Output: 3500

Program Output: 3500

Test Funtion: void withdrawAmount()

Test 1: Entering a value greater that customer’s current balance

```
Customer customer;
currentAmount = customer.getAmount();
int amountToWithdraw = 100000;
customer.withdrawAmount();

if((currentAmount < amountToWithdraw) && (customer.getAmount() != currentAmount)){
```

```
    cout << "Test 1 Failed" << endl;
}
```

User Input: 1000

Current Amount: 500

Expected Output: 500

Program Output: 500

Test 2: Entering negative value. If a negative amount is entered, program must not withdraw the amount

```
Customer customer;
currentAmount = customer.getAmount();
int amountToWithdraw = -100;
customer.withdrawAmount();

if((amountToWithdraw < 0) && (customer.getAmount() != currentAmount)){
    cout << "Test 2 Failed" << endl;
}
```

User Input: -1000

Initial Amount: 3500

Expected Output: 3500

Program Output: 3500

Test Funtion: void updateEmail(int accountNumber)

Test 1: Entering new email and checking if the email has been updated

```
Customer customer;
string oldEmail = customer.getEmail();
string newEmail = "new@email.com";
int accountNumber = 456326;
customer.updateEmail(accountNumber);

if(oldEmail == customer.getEmail()){
    cout << "Test Failed" << endl;
}
```

User Input: new@email.com

Old Email: abc@email.com

Expected Output: new@email.com

Program Output: new@email.com

Test Funtion: void updateContactNumber(int accountNumber)

Test 1: Entering new email and checking if the email has been updated

```
Customer customer;
string oldContactNumber = customer.getContactNumber();
string newContactNumber = "+61412345678";
int accountNumber = 456326;
customer.updateEmail(accountNumber);

if(oldContactNumber == customer.getContactNumber()){
    cout << "Test Failed" << endl;
}
```

User Input: +61412345678

Old Email: +61987654321

Expected Output: +61412345678

Program Output: +61412345678

Admin

Test Funtion: void updateEmail(int accountNumber)

Test 1: Entering new email and checking if the email has been updated

```
Admin admin;
string oldEmail = admin.getEmail();
string newEmail = "admin123@email.com";
int accountNumber = 587896;
admin.updateEmail(accountNumber);

if(oldEmail == admin.getEmail()){
    cout << "Test Failed" << endl;
}
```

User Input: admin123@email.com

Old Email: admin@email.com

Expected Output: admin123@email.com

Program Output: admin123@email.com

Test Funtion: void updateContactNumber(int accountNumber)

Test 1: Entering new contact number and checking if the number has been updated

```
Admin admin;
string oldContactNumber = admin.getContactNumber();
string newContactNumber = "+61475468315";
int accountNumber = 738020;
admin.updateContact(accountNumber);

if(oldContactNumber == admin.getContactNumber()){
    cout << "Test Failed" << endl;
}
```

User Input: +61475468315

Old Number: +61912121245

Expected Output: +61475468315

Program Output: +61475468315

Test Function: bool deleteAccount()

Test 1: Entering an account that exists and checking if it is deleted or not

```
Admin admin;
int accountToDelete = 738020;
bool flag = admin.deleteAccount();
int accountNumber = 738020;
if(!admin.deleteaccount()){ // delete accountNumer = 738020
  court<< "Test Failed" << endl;

admin.searchByAccount() //search accountNumber = 738020
```

User Input: 738020

User Input: 738020

Expected Output: Account deleted successfully
1
This account does not exist

Program Output: Account deleted successfully
1
This account does not exist

Test Function: void searchByName()

Test1: Entering a name of the account that exists

```
Admin admin;
string nameToSearch = "Customer1";
admin.searchByName(); // search nametoSearch = Customer1
```

User Input: Customer1

Expected Output: // All the account details related to the customer

Expected Output: // All the account details related to the customer

---

Tax Department

Test Funtion: double calculateTax()

Test 1: Checking if tax is calculated correctly

```
unsigned long long int totalAmountInBank = Bank::calculateTotalAmount();       // calling this function directly as it is a static fun
float taxRate = 0.15;
double tax = totalAmountInBank * 0.15;

TaxationDepartment taxationDepartment;
if(taxationDepartment.calculateTax() != tax){
    cout << "Test Failed" << endl;
}
```

Total Amount in bank: 20000000

Expected Output: **300000**

Program Output: **3000000**

SuperAdmin

Test Function: void deleteAdmin()

Test 1: Deleting an admin and then searching it if it exists

```
SuperAdmin superAdmin;
int adminToDelete = 145730;
superAdmin.deleteAdmin(); // delete adminToDelete  =145730
superAdmin.viewAdmins();
```

User Input: 145730

Expected Output: Account deleted successfully
//then displaying the info of the admin other than the deleted one

Program Output: Account deleted successfully
//then displaying the info of the admin other than the deleted one