CPSC 459 Blockchain Technologies
Project 1 – ScroogeCoin Autograder

Step 1. Copy **TxHandler.java** to **grading** folder.
Step 2. While holding down **Windows key** press **R** (or click **Start**) and then type **cmd**. Press **Enter**.
Step 3. Move into the folder that contains the **ScroogeCoin**-related classes by typing this command:

```
cd C:\grading
```

Step 4. Compile the classes with this command (also please note that the commands contained inside README are written for a *nix rig, that's why I replaced colons with semicolons):

```
javac -cp scroogeCoinGrader.jar;rsa.jar;algs4.jar;. TestTxHandler.java
```

Step 5. Finally, run the tests with this command:

```
java -cp scroogeCoinGrader.jar;rsa.jar;algs4.jar;. TestTxHandler
```

And here's entire output in the text format:

```
Running 7 total tests.

Test 1: test isValidTx() with valid transactions
==> passed

Test 2: test isValidTx() with transactions containing signatures of
incorrect data
==> passed

Test 3: test isValidTx() with transactions containing signatures using
incorrect private keys
==> passed

Test 4: test isValidTx() with transactions whose total output value
exceeds total input value
==> passed

Test 5: test isValidTx() with transactions that claim outputs not in the
current utxoPool
==> passed
```

```
Test 6: test isValidTx() with transactions that claim the same UTXO
multiple times
==> passed

Test 7: test isValidTx() with transactions that contain a negative output
value
==> passed


Total: 7/7 tests passed!

Running 8 total tests.

Test 1: test handleTransactions() with simple and valid transactions
Total Transactions = 2
Number of transactions returned valid by student = 2
Total Transactions = 50
Number of transactions returned valid by student = 50
Total Transactions = 100
Number of transactions returned valid by student = 100
==> passed

Test 2: test handleTransactions() with simple but some invalid
transactions because of invalid signatures
Total Transactions = 2
Number of transactions returned valid by student = 0
Total Transactions = 50
Number of transactions returned valid by student = 1
Total Transactions = 100
Number of transactions returned valid by student = 1
==> passed

Test 3: test handleTransactions() with simple but some invalid
transactions because of inputSum < outputSum
Total Transactions = 2
Number of transactions returned valid by student = 1
Total Transactions = 50
Number of transactions returned valid by student = 18
```

```
Total Transactions = 100
Number of transactions returned valid by student = 41
==> passed

Test 4: test handleTransactions() with simple and valid transactions with
some double spends
Total Transactions = 2
Number of transactions returned valid by student = 1
Total Transactions = 50
Number of transactions returned valid by student = 23
Total Transactions = 100
Number of transactions returned valid by student = 43
==> passed

Test 5: test handleTransactions() with valid but some transactions are
simple, some depend on other transactions
Total Transactions = 2
Number of transactions returned valid by student = 1
Total Transactions = 50
Number of transactions returned valid by student = 26
Total Transactions = 100
Number of transactions returned valid by student = 94
==> passed

Test 6: test handleTransactions() with valid and simple but some
transactions take inputs from non-exisiting utxo's
Total Transactions = 2
Number of transactions returned valid by student = 1
Total Transactions = 50
Number of transactions returned valid by student = 10
Total Transactions = 100
Number of transactions returned valid by student = 59
==> passed

Test 7: test handleTransactions() with complex Transactions
Total Transactions = 2
Number of transactions returned valid by student = 0
Total Transactions = 50
```

```
Number of transactions returned valid by student = 12
Total Transactions = 100
Number of transactions returned valid by student = 20
==> passed

Test 8: test handleTransactions() with simple, valid transactions being
called again to check for changes made in the pool
Total Transactions = 2
Number of transactions returned valid by student = 2
Total Transactions = 50
Number of transactions returned valid by student = 49
Total Transactions = 100
Number of transactions returned valid by student = 46
==> passed



Total: 8/8 tests passed!
```

Testing MaxFeeTxHandler.java: Enter the following commands in your terminal from the current working directory.

```
$javac -cp scroogeCoinGrader.jar;rsa.jar;algs4.jar;. TestMaxFeeTxHandler.java
$java -cp scroogeCoinGrader.jar;rsa.jar;algs4.jar;. TestMaxFeeTxHandler
```

Note: If you are using IDE like eclipse, just add jar files to your build path and create a folder for all text files within the working directory of your code.

Jar Files:
1) rsa.jar: Contains classes for using RSAKeys
2) algs4.jar: Contains some useful classes like defining priority queues, stacks, etc.
3) scroogeCoinGrader.java: Contains classes used for grading the submitted files.