CSCI361 Cryptography – Assignment 2
Name : LEE CHIA LIN
Student ID : 6097881

Task One – RSA

1. $N = 59 * 47 = 2773$
   Public key: (e, N) = (15, 2773)

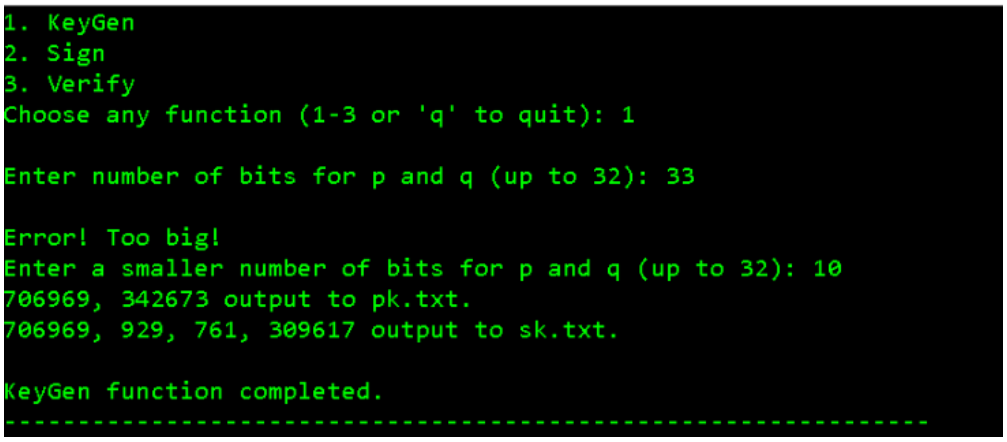
2. $(59 - 1) * (47 - 1) = 2668$
   $d = e^{-1}\ mod\ 2668 = 15^{-1}\ mod\ 2668$

   $\gcd(2668, 15) = 1$

n1	n2	r	q	a1	b1	a2	b2
2668	15	13	177	1	0	0	1
15	13	2	1	0	1	1	-177
13	2	1	6	1	-177	-1	178
2	1	0	2	-1	178	7	-1245

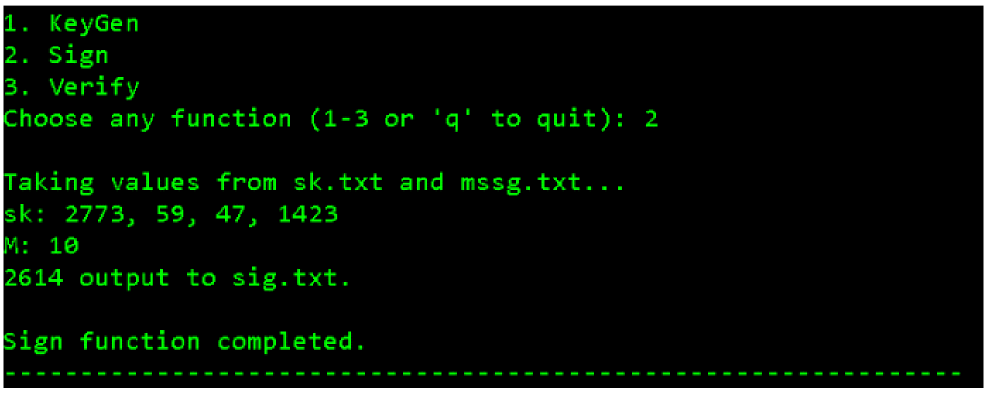
   $1 = (7 * 2668) + (-1245 * 15)$
   $d = -1245 + 2668 = 1423$
   Private key: (d, N) = (1423, 2773)

3, 4 and 5.
rsa.java
Runtime SAMPLE - KeyGen



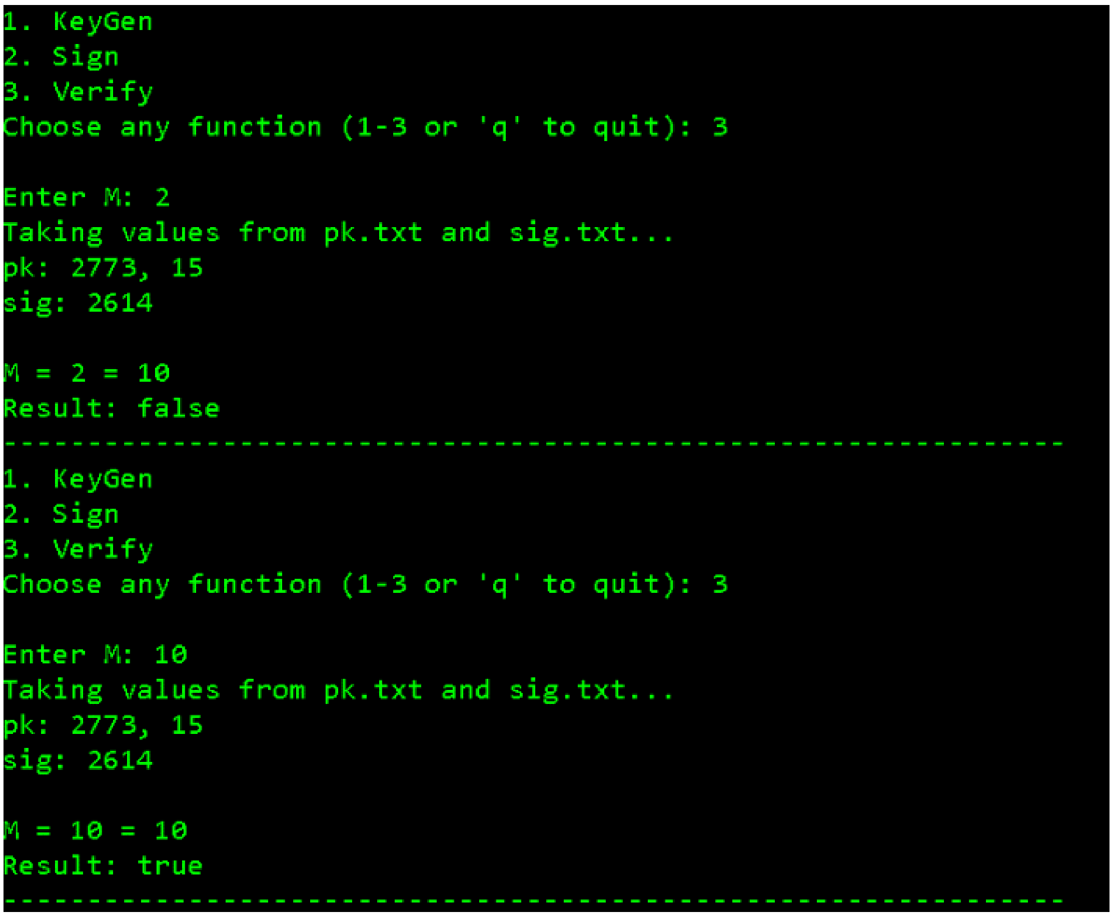
```
1. KeyGen
2. Sign
3. Verify
Choose any function (1-3 or 'q' to quit): 1

Enter number of bits for p and q (up to 32): 33

Error! Too big!
Enter a smaller number of bits for p and q (up to 32): 10
706969, 342673 output to pk.txt.
706969, 929, 761, 309617 output to sk.txt.

KeyGen function completed.
---------------------------------------------------------------
```

Runtime SAMPLE - Sign



```
1. KeyGen
2. Sign
3. Verify
Choose any function (1-3 or 'q' to quit): 2

Taking values from sk.txt and mssg.txt...
sk: 2773, 59, 47, 1423
M: 10
2614 output to sig.txt.

Sign function completed.
---------------------------------------------------------------
```

Runtime SAMPLE - Verify



```
1. KeyGen
2. Sign
3. Verify
Choose any function (1-3 or 'q' to quit): 3

Enter M: 2
Taking values from pk.txt and sig.txt...
pk: 2773, 15
sig: 2614

M = 2 = 10
Result: false
---------------------------------------------------------------
1. KeyGen
2. Sign
3. Verify
Choose any function (1-3 or 'q' to quit): 3

Enter M: 10
Taking values from pk.txt and sig.txt...
pk: 2773, 15
sig: 2614

M = 10 = 10
Result: true
---------------------------------------------------------------
```

<u>Manual calculation for RSA 4. and 5.</u>

$\boldsymbol{Sign\ M = 10}$

$sig = M^d \ mod\ N = \ 10^{1423} \ mod\ 2773$

$1423\ (decimal) = 10110001111\ (binary)$

$S: 10^2 \qquad mod\ 2773 = 100$

$SX: 100^2 * 10\ mod\ 2773 = 172$

$SX: 172^2 * 10\ mod\ 2773 = 1902$

$S: 1902^2 \qquad mod\ 2773 = 1612$

$S: 1612^2 \qquad mod\ 2773 = 243$

$S: 243^2 \qquad mod\ 2773 = 816$

$SX: 816^2 * 10\ mod\ 2773 = 587$

$SX: 587^2 * 10\ mod\ 2773 = 1624$

$SX: 1624^2 * 10\ mod\ 2773 = 2530$

$SX: 2530^2 * 10\ mod\ 2773 = \mathbf{2614}$

$\boldsymbol{sig = 2614}$

$\boldsymbol{Verify}$

$M = sig^e \, mod \, N = 2614^{15} \, mod \, 2773$

$15 \, (decimal) = 1111 \, (binary)$

$SX: 2614^2 * 2614 \, mod \, 2773 = 1171$

$SX: 1171^2 * 2614 \, mod \, 2773 = 2579$

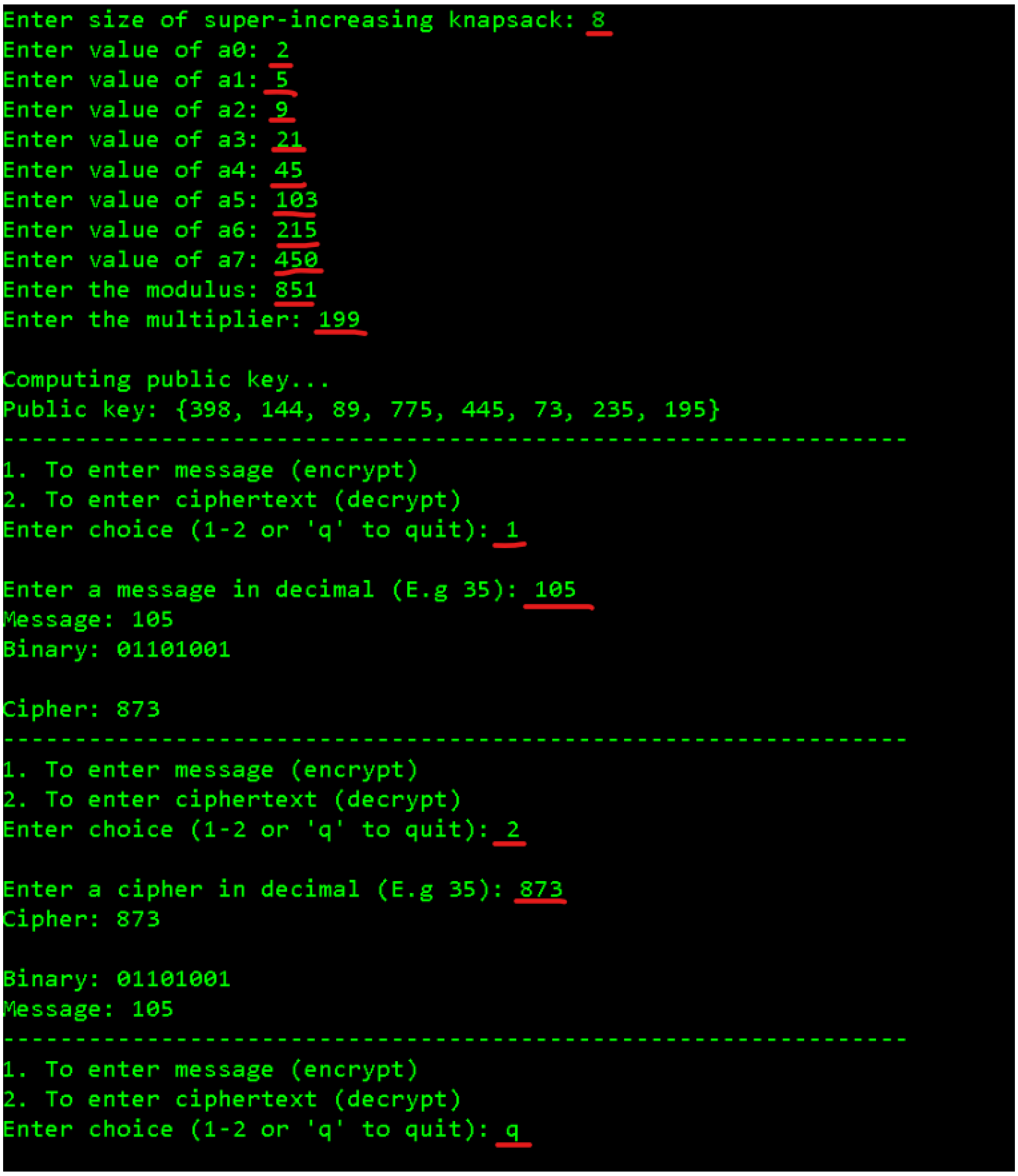
$SX: 2579^2 * 2614 \, mod \, 2773 = \mathbf{10}$

$\boldsymbol{M} = \mathbf{10}$

Task Two – Knapsack

knapsack.java

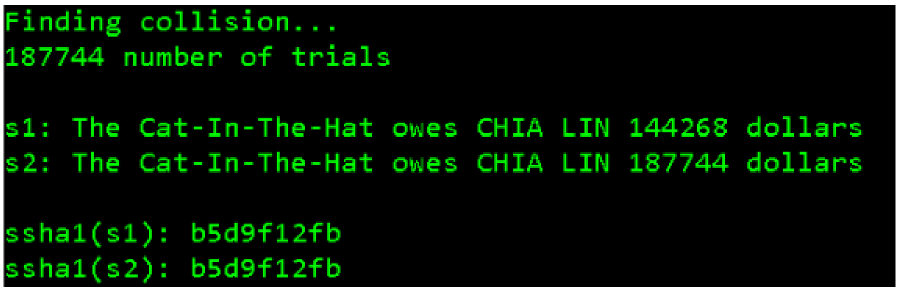
User input are underline in **RED**



```
Enter size of super-increasing knapsack: 8
Enter value of a0: 2
Enter value of a1: 5
Enter value of a2: 9
Enter value of a3: 21
Enter value of a4: 45
Enter value of a5: 103
Enter value of a6: 215
Enter value of a7: 450
Enter the modulus: 851
Enter the multiplier: 199

Computing public key...
Public key: {398, 144, 89, 775, 445, 73, 235, 195}
---------------------------------------------------------------
1. To enter message (encrypt)
2. To enter ciphertext (decrypt)
Enter choice (1-2 or 'q' to quit): 1

Enter a message in decimal (E.g 35): 105
Message: 105
Binary: 01101001

Cipher: 873
---------------------------------------------------------------
1. To enter message (encrypt)
2. To enter ciphertext (decrypt)
Enter choice (1-2 or 'q' to quit): 2

Enter a cipher in decimal (E.g 35): 873
Cipher: 873

Binary: 01101001
Message: 105
---------------------------------------------------------------
1. To enter message (encrypt)
2. To enter ciphertext (decrypt)
Enter choice (1-2 or 'q' to quit): q
```

Task Three – Collision Finding of Hash functions

collision.java

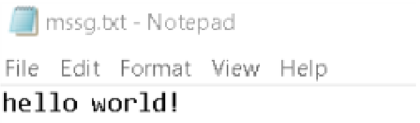


```
Finding collision...
187744 number of trials

s1: The Cat-In-The-Hat owes CHIA LIN 144268 dollars
s2: The Cat-In-The-Hat owes CHIA LIN 187744 dollars

ssha1(s1): b5d9f12fb
ssha1(s2): b5d9f12fb
```

Task Four – DSA

dsa.java

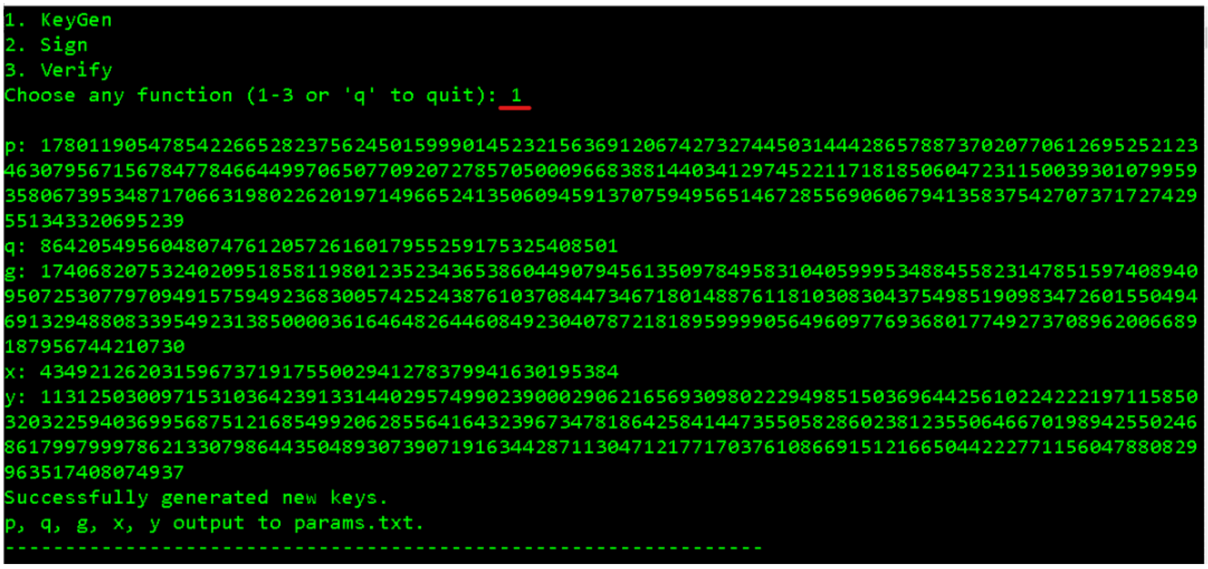
SAMPLE input file mssg.txt



```
hello world!
```

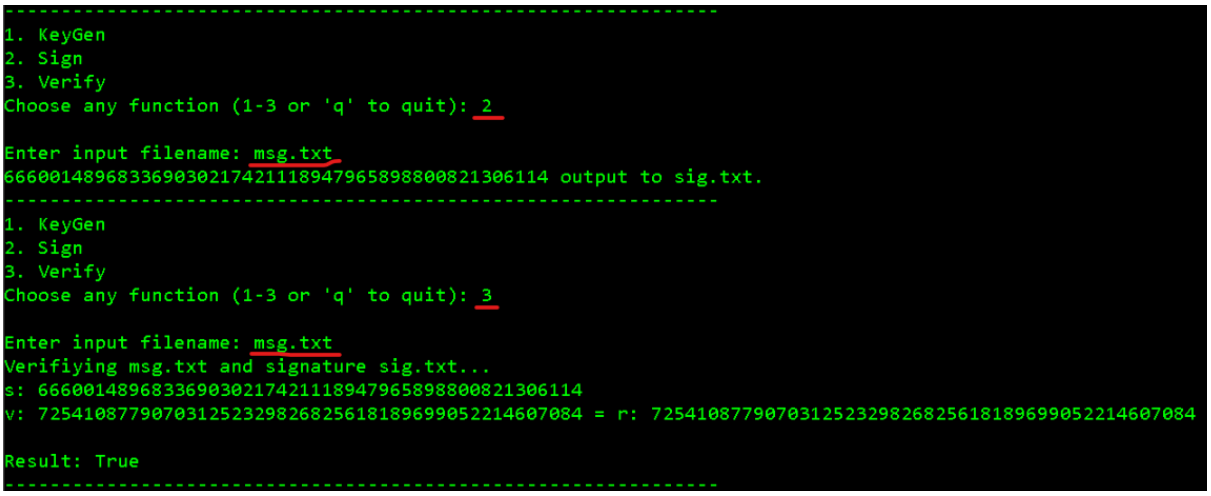
User input are underline in **RED**

KeyGen



```
1. KeyGen
2. Sign
3. Verify
Choose any function (1-3 or 'q' to quit): 1

p: 178011905478542266528237562450159990145232156369120674273274450314442865788737020770612695252123
463079567156784778466449970650770920727857050009668388144034129745221171818506047231150039301079959
358067395348717066319802262019714966524135060945913707594956514672855690606794135837542707371727429
551343320695239
q: 864205495604807476120572616017955259175325408501
g: 174068207532402095185811980123523436538604490794561350978495831040599953488455823147851597408940
950725307797094915759492368300574252438761037084473467180148876118103083043754985190983472601550494
691329488083395492313850000361646482644608492304078721818959999056496097769368017749273708962006689
187956744210730
x: 434921262031596737191755002941278379941630195384
y: 113125030097153103642391331440295749902390002906216569309802229498515036964425610224222197115850
320322594036995687512168549920628556416432396734781864258414473550582860238123550646670198942550246
861799799978621330798644350489307390719163442871130471217717037610866915121665044222771156047880829
963517408074937
Successfully generated new keys.
p, q, g, x, y output to params.txt.
---------------------------------------------------------------
```

Sign and Verify



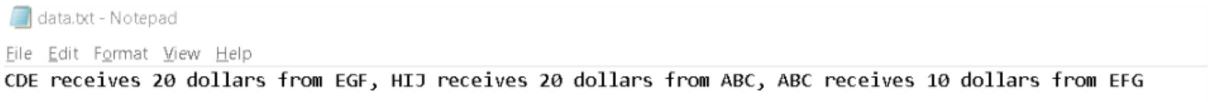
```
------------------------------------------------------------
1. KeyGen
2. Sign
3. Verify
Choose any function (1-3 or 'q' to quit): 2

Enter input filename: msg.txt
666001489683369030217421118947965898800821306114 output to sig.txt.
------------------------------------------------------------
1. KeyGen
2. Sign
3. Verify
Choose any function (1-3 or 'q' to quit): 3

Enter input filename: msg.txt
Verifiying msg.txt and signature sig.txt...
s: 666001489683369030217421118947965898800821306114
v: 725410877907031252329826825618189699052214607084 = r: 725410877907031252329826825618189699052214607084

Result: True
------------------------------------------------------------
```

Task Five – hashcash

hashcash.java

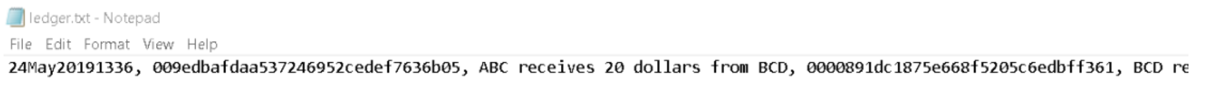
.txt information are saved in the format using ", " as delimiter

SAMPLE **data.txt**



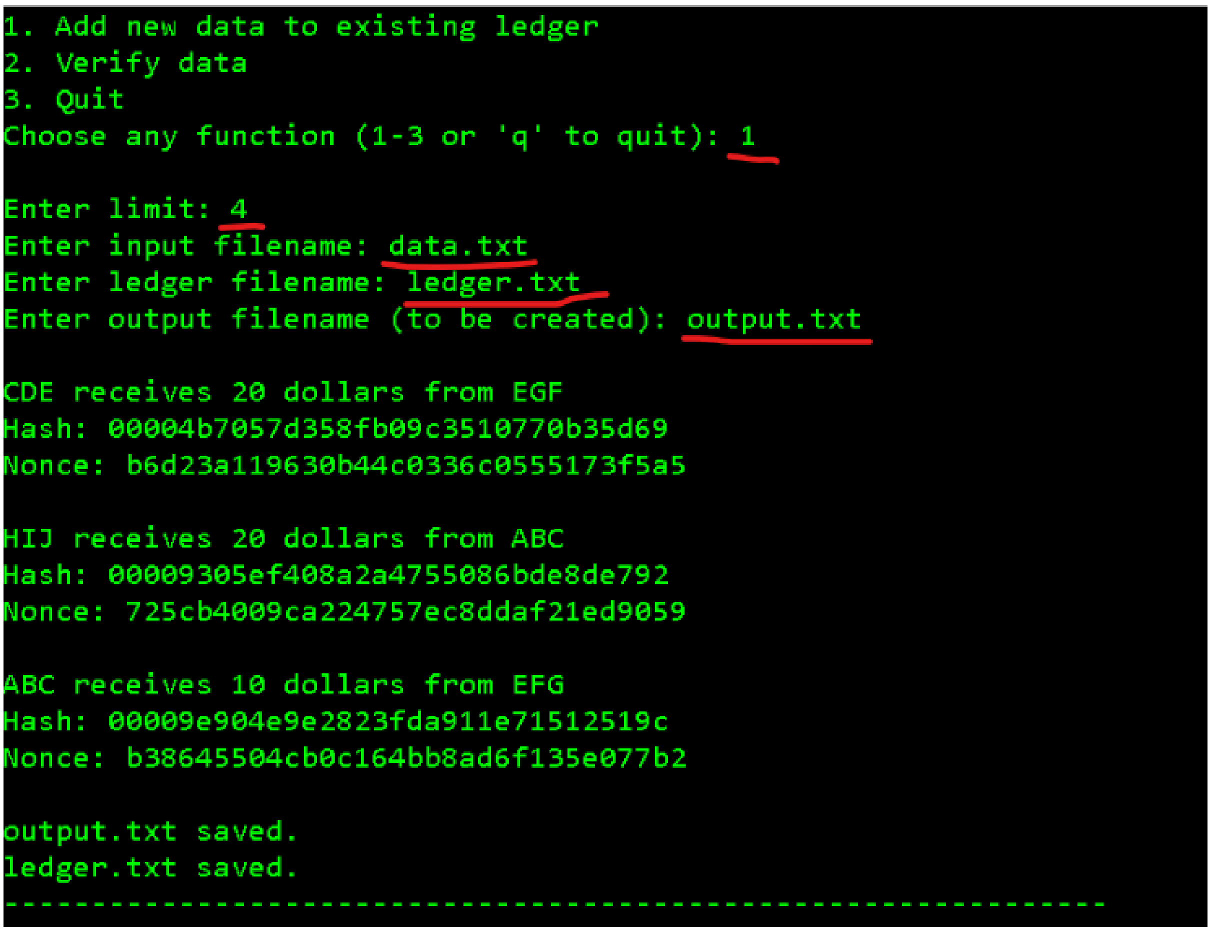
```
CDE receives 20 dollars from EGF, HIJ receives 20 dollars from ABC, ABC receives 10 dollars from EFG
```

SAMPLE **ledger.txt**



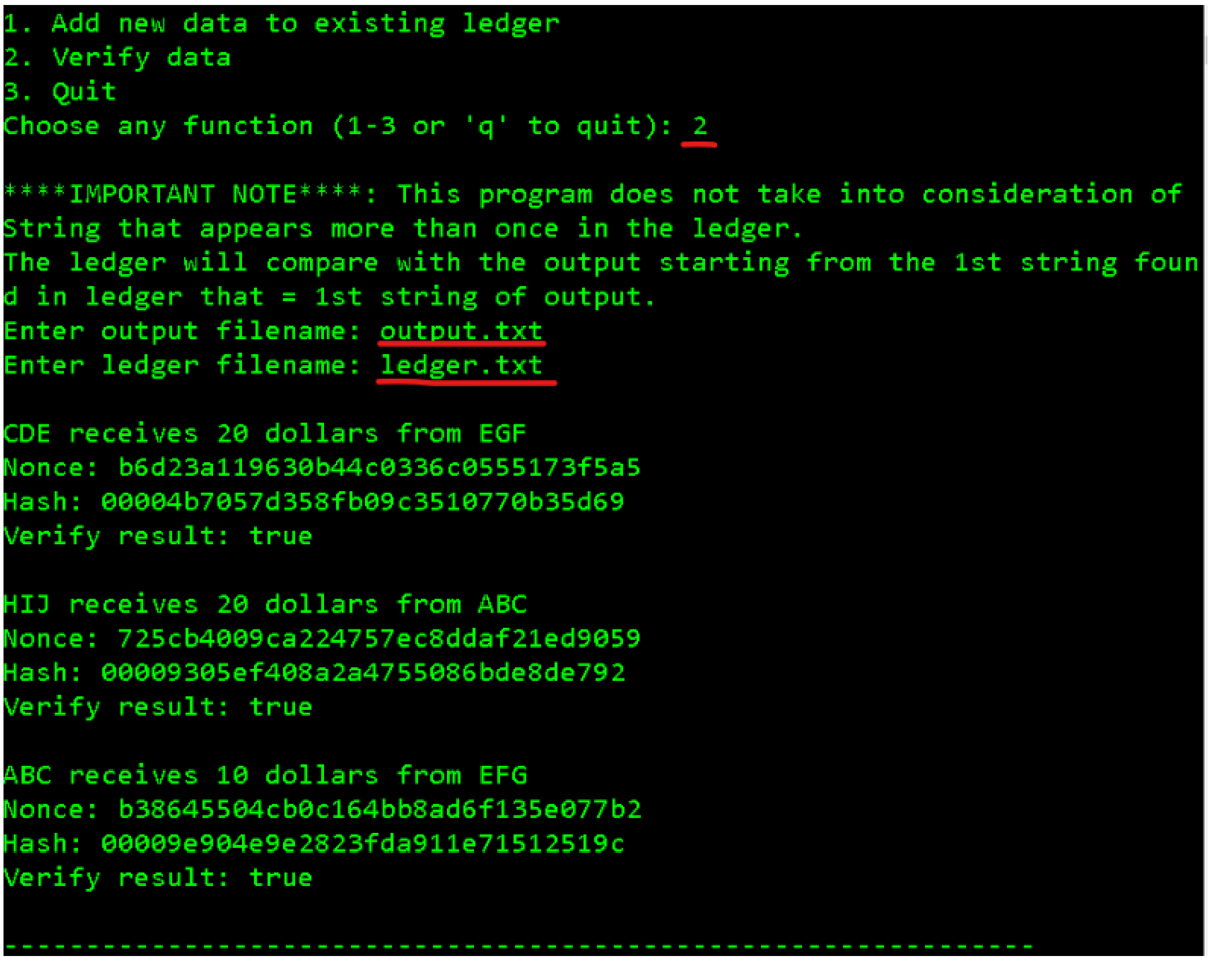
```
24May20191336, 009edbafdaa537246952cedef7636b05, ABC receives 20 dollars from BCD, 0000891dc1875e668f5205c6edbff361, BCD re
```

Add **data.txt to ledger.txt**



```
1. Add new data to existing ledger
2. Verify data
3. Quit
Choose any function (1-3 or 'q' to quit): 1

Enter limit: 4
Enter input filename: data.txt
Enter ledger filename: ledger.txt
Enter output filename (to be created): output.txt

CDE receives 20 dollars from EGF
Hash: 00004b7057d358fb09c3510770b35d69
Nonce: b6d23a119630b44c0336c0555173f5a5

HIJ receives 20 dollars from ABC
Hash: 00009305ef408a2a4755086bde8de792
Nonce: 725cb4009ca224757ec8ddaf21ed9059

ABC receives 10 dollars from EFG
Hash: 00009e904e9e2823fda911e71512519c
Nonce: b38645504cb0c164bb8ad6f135e077b2

output.txt saved.
ledger.txt saved.
-------------------------------------------------------------
```

Verify/Validate **output.txt and ledger.txt**



```
1. Add new data to existing ledger
2. Verify data
3. Quit
Choose any function (1-3 or 'q' to quit): 2

****IMPORTANT NOTE****: This program does not take into consideration of
String that appears more than once in the ledger.
The ledger will compare with the output starting from the 1st string foun
d in ledger that = 1st string of output.
Enter output filename: output.txt
Enter ledger filename: ledger.txt

CDE receives 20 dollars from EGF
Nonce: b6d23a119630b44c0336c0555173f5a5
Hash: 00004b7057d358fb09c3510770b35d69
Verify result: true

HIJ receives 20 dollars from ABC
Nonce: 725cb4009ca224757ec8ddaf21ed9059
Hash: 00009305ef408a2a4755086bde8de792
Verify result: true

ABC receives 10 dollars from EFG
Nonce: b38645504cb0c164bb8ad6f135e077b2
Hash: 00009e904e9e2823fda911e71512519c
Verify result: true

---------------------------------------------------------------
```