SEM - VII - 2022-23

CNS Lab

B3 - 2019BTECS00094 - Sweety Shrawan Gupta

Assignment 8

Euclidean,Extended Euclidean,Multiplicative Inverse

Theory:-

Euclidean

The **Euclidean Algorithm** is a technique for quickly finding the **GCD** of two integers.

The Algorithm

The Euclidean Algorithm for finding GCD(A,B) is as follows:

- If A = 0 then GCD(A,B)=B, since the GCD(0,B)=B, and we can stop.
- If B = 0 then GCD(A,B)=A, since the GCD(A,0)=A, and we can stop.
- Write A in quotient remainder form (A = B·Q + R)
- Find GCD(B,R) using the Euclidean Algorithm since GCD(A,B) = GCD(B,R)

Extended Euclidean

In arithmetic and computer programming, the extended Euclidean algorithm is an extension to the Euclidean algorithm, and computes, in addition to the greatest common divisor (gcd) of integers a and b, also the coefficients of Bézout's identity, which are integers x and y such that

Multiplicative Inverse

A modular multiplicative inverse of an integer a is an integer x such that $a \cdot x$ is congruent to 1 modular some modulus m. To write it in a formal way: we want to find an integer x so that

$$a \cdot x \equiv 1 \mod m.$$

We will also denote x simply with a^{-1}.

We should note that the modular inverse does not always exist. For example, let $m = 4$, $a = 2$. By checking all possible values modulo m is should become clear that we cannot find a^{-1} satisfying the above equation. It can be proven that the modular inverse exists if and only if a and m are relatively prime (i.e. $\gcd(a, m) = 1$).

We use Extended Euclidean Algorithm to find the solution.

Euclidean

Code:

Cpp:

```cpp
#include <bits/stdc++.h>
using namespace std;

int main()
{

    int q, r1, r2, r;
    cin >> r1 >> r2;
    cout << r1 << " " << r2 << "\n";
    while (r2 > 0)
    {
        q = r1 / r2;
        r = r1 - r2 * q;
        r1 = r2;
        r2 = r;
    }
    cout << r1 << "\n";

    return 0;
}
```

```
d:\CNS Lab>cd "d:\CNS Lab\" && g++ euclidean.cpp -o euclidean && "d:\CNS Lab\"euclidean
4 8
4 8
4
```

Python:

```python
def euclidean(a: int, b: int):
  r1 = a
  r2 = b

  print(f'{"q"} {"r1":>20} {"r2":>20} {"r":>20}')
  print("")

  while r2 != 0:
    q = r1 // r2
    r = r1 - q * r2
    print(f'{q} {"%20.0d" % r1} {"%20.0d" % r2} {"%20.0d" % r}')

    r1 = r2
    r2 = r

  print(f'{"-"} {"%20.0d" % r1} {"%20.0d" % r2} {"-":>20}')

  print(f'GCD({a}, {b}) = {r1}')




def main():

  a = int(input('Enter a: '))
  b = int(input('Enter b: '))
  euclidean(a,b)




main()
```

Output:

```
In [1]: runfile('D:/CNS Lab/euclidean.py', wdir='D:/CNS Lab')

Enter a: 56738901534

Enter b: 45778901246
q                   r1                   r2                    r

1          56738901534          45778901246          10960000288
4          45778901246          10960000288           1938900094
5          10960000288           1938900094           1265499818
1           1938900094           1265499818            673400276
1           1265499818            673400276            592099542
1            673400276            592099542             81300734
7            592099542             81300734             22994404
3             81300734             22994404             12317522
1             22994404             12317522             10676882
1             12317522             10676882              1640640
6             10676882              1640640               833042
1              1640640               833042               807598
1               833042               807598                25444
31               807598                25444                18834
1                 25444                18834                 6610
2                 18834                 6610                 5614
1                  6610                 5614                  996
5                  5614                  996                  634
1                   996                  634                  362
1                   634                  362                  272
1                   362                  272                   90
3                   272                   90                    2
45                   90                    2                    0
-                     2                    0                    -
GCD(56738901534, 45778901246) = 2
```

```
In [4]: runfile('D:/CNS Lab/euclidean.py', wdir='D:/CNS Lab')

Enter a: 07512779912565014928652150296195898826835317022090

Enter b: 97569354727979857921728118865439213176360669242153
q                  r1                  r2                  r

0 75127799125650149286521502961958988268353170220907 97569354727979857921728118865439213176360669242153 75127799125650149286521502961958988268353170220907
12 97569354727979857921728118865439213176360669242153 75127799125650149286521502961958988268353170220907 74159957771996787779023153110884272543368649770737
1 75127799125650149286521502961958988268353170220907 74159957771996787779023153110884272543368649770737 96784135365336150749834985107471572498452045017
76 74159957771996787779023153110884272543368649770737 96784135365336150749834985107471572498452045017 60401489434131320914856442920587744454509555781
1 96784135365336150749834985107471572498452045017 60401489434131320914856442920587744454509555781 36382645931204829834978542186883828043942489236
1 60401489434131320914856442920587744454509555781 36382645931204829834978542186883828043942489236 24018843502926491079877900733703916410567066545
1 36382645931204829834978542186883828043942489236 24018843502926491079877900733703916410567066545 12363802428278338755100641453179911633375422691
1 24018843502926491079877900733703916410567066545 12363802428278338755100641453179911633375422691 11655041074648152324777259280524004777191643854
1 12363802428278338755100641453179911633375422691 11655041074648152324777259280524004777191643854 70876135363018643032338217265590685618377883 7
16 11655041074648152324777259280524004777191643854 70876135363018643032338217265590685618377883 7 31485941656516943960314451802949507825118246 2
2 70876135363018643032338217265590685618377883 7 31485941656516943960314451802949507825118246 2 79042520499847551117093136596916699681413913
3 31485941656516943960314451802949507825118246 2 79042520499847551117093136596916699681413913 7 77318550656267862518651082387449792069407 23
1 79042520499847551117093136596916699681413913 77731855065626786251865108238744979206940723 13106654342207648652280283581717204744731 90
59 77731855065626786251865108238744979206940723 13106654342207648652280283581717204744731 90 40259444660165920341143510661347121302251 3
3 13106654342207648652280283581717204744731 90 40259444660165920341143510661347121302251 3 10288209441578725499372303833130683540565 1
3 40259444660165920341143510661347121302251 3 10288209441578725499372303833130683540565 1 93948163354297438430265991619550706805560 8
1 10288209441578725499372303833130683540565 1 93948163354297438430265991619550706805560 8 93393106148981656345704671175612860009 1
10 93948163354297438430265991619550706805560 8 93393106148981656345704671175612860009 1 46088527393992727956955245019894208040650 4
1 8933931061489816563457046711756128600091 4608852739399272795695524501989420804650 1 43250783220905437677615222097667077954 41
1 4608852739399272795695524501989420804650 1 43250783220905437677615222097667077954 41 28377441730872902793400229222271300920 9
15 43250783220905437677615222097667077954 41 28377441730872902793400229222271300920 9 68462062459608348751487826426012657306 9
4 28377441730872902793400229222271300920 9 68462062459608348751487826426012657306 9 99261674702956329280509865186623799 8
6 68462062459608348751487826426012657306 9 99261674702956329280509865186623799 8 89050576378345511831819071314038377396 1
1 99261674702956329280509865186623799 8 89050576378345511831819071314038377396 1 10211098324610817448690792046240025 89
8 89050576378345511831819071314038377396 1 10211098324610817448690792046240025 89 73617897814589722422927367704635668 4
1 10211098324610817448690792046240025 89 73617897814589722422927367704635668 4 28493085431518452063980552757764590 5
2 73617897814589722422927367704635668 4 28493085431518452063980552757764590 5 16631726951552818294966262189106487 4
1 28493085431518452063980552757764590 5 16631726951552818294966262189106487 4 11861358479965633769014290568658103 1
```

```
1 1291253980046550917312769649257153 125489770712128157678638924024174 4 3635627292526934052638040901540 9
34 125489770712128157678638924024174 4 3635627292526934052638040901540 9 1878442766212399888945533371783 8
1 3635627292526934052638040901540 9 1878442766212399888945533371783 8 1757184526314534163692507529757 1
1 1878442766212399888945533371783 8 1757184526314534163692507529757 1 1212582398978657252530258420267
14 1757184526314534163692507529757 1 1212582398978657252530258420267 5956916774441401015014574138 3
2 1212582398978657252530258420267 5956916774441401015014574138 3 2119904409037704952734359260 1
28 5956916774441401015014574138 3 2119904409037704952734359260 1 2118442913582714735836821005 1
10 2119904409037704952734359260 1 2118442913582714735836821005 1 1461495454990216897538255 1
144 2118442913582714735836821005 1 1461495454990216897538255 1 1388945839680240338173366 1
1 1461495454990216897538255 1 1388945839680240338173366 1 7254961530997655936488 90
19 1388945839680240338173366 1 7254961530997655936488 90 1050314879068571024047 51
6 7254961530997655936488 90 1050314879068571024047 51 9530722565862297922038 4
1 1050314879068571024047 51 9530722565862297922038 4 9724262248234123184367
9 9530722565862297922038 4 9724262248234123184367 7788865424515870561081
1 9724262248234123184367 7788865424515870561081 1935396823718252623286
4 7788865424515870561081 1935396823718252623286 4727812964286006793 7
40 1935396823718252623286 4727812964286006793 7 4427163800384990581 6
1 4727812964286006793 7 4427163800384990581 6 3006491639010162131
14 4427163800384990581 6 3006491639010162131 2 1807550577076359 7
1 6 3006491639010162131 2 1807550577076359 7 8257365813025261 59
2 2 1807550577076359 7 8257365813025261 59 2928189510258365 4
1 7 8257365813025261 59 2928189510258365 4 2964546861999425 5
1 59 2928189510258365 4 2964546861999425 5 2328272089026411 49
1 4 2964546861999425 5 2328272089026411 49 6362747729730135 6
3 5 2328272089026411 49 6362747729730135 6 4194477701073708 1
1 49 6362747729730135 6 4194477701073708 1 2168270028656427 5
1 6 4194477701073708 1 2168270028656427 5 2026207672417280 6
1 1 2168270028656427 5 2026207672417280 6 142062356239146 9
14 5 2026207672417280 6 142062356239146 9 37334685069224 0
3 6 142062356239146 9 37334685069224 0 3005830103147 49
1 9 37334685069224 0 3005830103147 49 7276384037749 1
4 0 3005830103147 49 7276384037749 1 952764880478 5
7 49 7276384037749 1 952764880478 5 607029874399 6
1 1 952764880478 5 607029874399 6 345735006078 9
1 5 607029874399 6 345735006078 9 261294868320 7
1 6 345735006078 9 261294868320 7 84440137758 2
3 9 261294868320 7 84440137758 2 9744550461
```

```
1       373346850692240       300583010314749       72763840377491
4       300583010314749       72763840377491        9527648804785
7       72763840377491        9527648804785         6070298743996
1       9527648804785         6070298743996         3457350060789
1       6070298743996         3457350060789         2612948683207
1       3457350060789         2612948683207         844401377582
3       2612948683207         844401377582          79744550461
10      844401377582          79744550461           46955872972
1       79744550461           46955872972           32788677489
1       46955872972           32788677489           14167195483
2       32788677489           14167195483           4454286523
3       14167195483           4454286523            804335914
5       4454286523            804335914             432606953
1       804335914             432606953             371728961
1       432606953             371728961             60877992
6       371728961             60877992              6461009
9       60877992              6461009               2728911
2       6461009               2728911               1003187
2       2728911               1003187               722537
1       1003187               722537                280650
2       722537                280650                161237
1       280650                161237                119413
1       161237                119413                41824
2       119413                41824                 35765
1       41824                 35765                 6059
5       35765                 6059                  5470
1       6059                  5470                  589
9       5470                  589                   169
3       589                   169                   82
2       169                   82                    5
16      82                    5                     2
2       5                     2                     1
2       2                     1                     0
-       1                     0                     -
GCD(751277991256501492865215029619589882683531702209O, 97569354727979857921728118865439213176360669242153) = 1
```

Extended Euclidean

Code:

Cpp:

```cpp
#include <bits/stdc++.h>
using namespace std;

int main()
{

    int q, r1, r2, r, s1 = 1, s2 = 0, s, t1 = 0, t2 = 1, t;
    cin >> r1 >> r2;
    cout << r1 << " " << r2 << "\n";
    while (r2 > 0)
    {
        q = r1 / r2;
        r = r1 - r2 * q;
        r1 = r2;
```

```
        r2 = r;
        s = s1 - s2 * q;
        s1 = s2;
        s2 = s;
        t = t1 - t2 * q;
        t1 = t2;
        t2 = t;
    }
    cout << r1 << " " << s1 << " " << t1 << "\n";

    return 0;
}
```

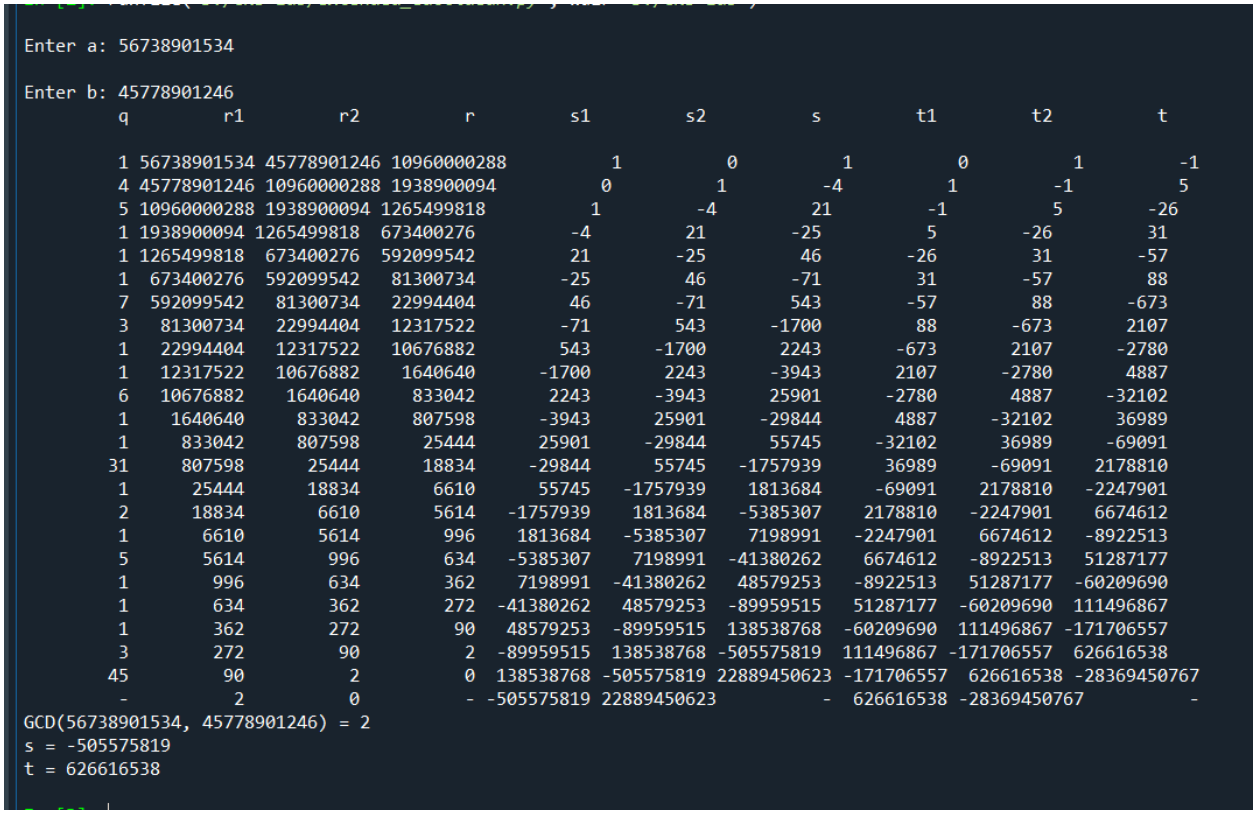
Output:

```
d:\CNS Lab>cd "d:\CNS Lab\" && g++ ext_euclidean.cpp -o ext_euclidean ean && "d:\CNS Lab\"ext_
ean && "d:\CNS Lab\"ext_euclidean
56 32
56 32
8 -1 2
```

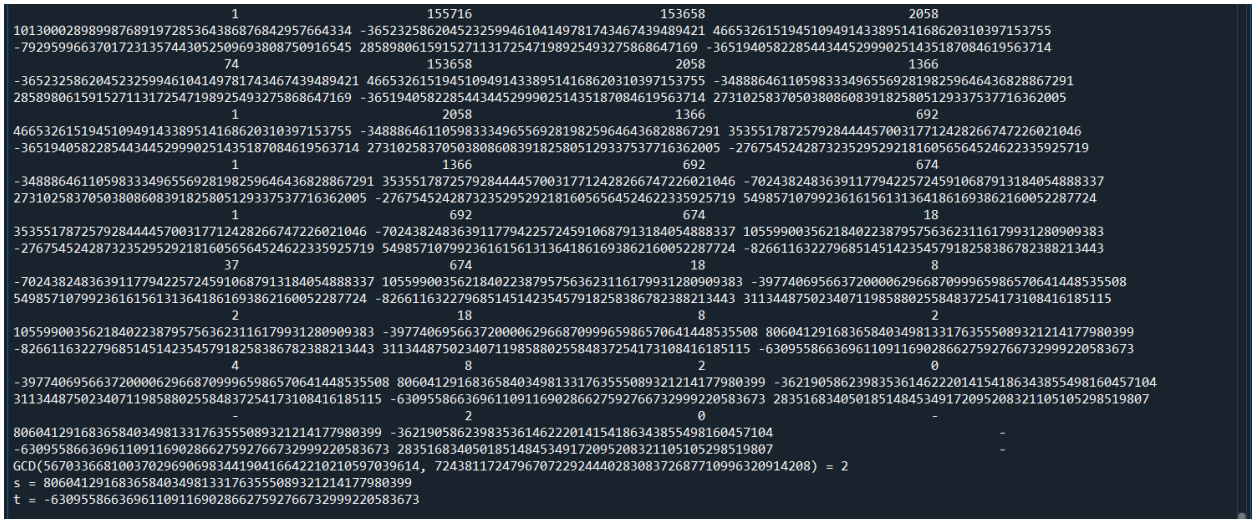
Python:

```
def extEuclidean(a: int, b: int):
  r1 = a
  r2 = b
  s1 = 1
  s2 = 0
  t1 = 0
  t2 = 1

  print(f'{"q":>10} {"r1":>10} {"r2":>10} {"r":>10} {"s1":>10} {"s2":>10}
{"s":>10} {"t1":>10} {"t2":>10} {"t":>10}')
  print(" ")
  while r2 != 0:
    q = r1 // r2
    r = r1 - q * r2
    s = s1 - q * s2
    t = t1 - q * t2
```

```
    print(f'{"%10.0d" % q} {"%10.0d" % r1} {"%10.0d" % r2} {"%10.0d" % r}
{"%10.0d" % s1} {"%10.0d" % s2} {"%10.0d" % s} {"%10.0d" % t1} {"%10.0d" %
t2} {"%10.0d" % t}')

    r1 = r2
    r2 = r
    s1 = s2
    s2 = s
    t1 = t2
    t2 = t

  print(f'{"-":>10} {"%10.0d" % r1} {"%10.0d" % r2} {"-":>10} {"%10.0d" %
s1} {"%10.0d" % s2} {"-":>10} {"%10.0d" % t1} {"%10.0d" % t2} {"-":>10}')

  print(f'GCD({a}, {b}) = {r1}\ns = {s1}\nt = {t1}')



def main():

  a = int(input('Enter a: '))
  b = int(input('Enter b: '))
  extEuclidean(a,b)




main()
```



```
Enter a: 56738901534

Enter b: 45778901246
        q         r1         r2          r         s1         s2          s         t1         t2          t

        1 56738901534 45778901246 10960000288          1          0          1          0          1         -1
        4 45778901246 10960000288 1938900094          0          1         -4          1         -1          5
        5 10960000288 1938900094 1265499818          1         -4         21         -1          5        -26
        1 1938900094 1265499818  673400276         -4         21        -25          5        -26         31
        1 1265499818  673400276  592099542         21        -25         46        -26         31        -57
        1  673400276  592099542   81300734        -25         46        -71         31        -57         88
        7  592099542   81300734   22994404         46        -71        543        -57         88       -673
        3   81300734   22994404   12317522        -71        543      -1700         88       -673       2107
        1   22994404   12317522   10676882        543      -1700       2243       -673       2107      -2780
        1   12317522   10676882    1640640      -1700       2243      -3943       2107      -2780       4887
        6   10676882    1640640     833042       2243      -3943      25901      -2780       4887     -32102
        1    1640640     833042     807598      -3943      25901     -29844       4887     -32102      36989
        1     833042     807598      25444      25901     -29844      55745     -32102      36989     -69091
       31     807598      25444      18834     -29844      55745   -1757939      36989     -69091    2178810
        1      25444      18834       6610      55745   -1757939    1813684     -69091    2178810   -2247901
        2      18834       6610       5614   -1757939    1813684   -5385307    2178810   -2247901    6674612
        1       6610       5614        996    1813684   -5385307    7198991   -2247901    6674612   -8922513
        5       5614        996        634   -5385307    7198991  -41380262    6674612   -8922513   51287177
        1        996        634        362    7198991  -41380262   48579253   -8922513   51287177  -60209690
        1        634        362        272  -41380262   48579253  -89959515   51287177  -60209690  111496867
        1        362        272         90   48579253  -89959515  138538768  -60209690  111496867 -171706557
        3        272         90          2  -89959515  138538768 -505575819  111496867 -171706557  626616538
       45         90          2          0  138538768 -505575819 22889450623 -171706557  626616538 -28369450767
        -          2          0          - -505575819 22889450623          -  626616538 -28369450767          -
GCD(56738901534, 45778901246) = 2
s = -505575819
t = 626616538
```



```
                    1                    155716                    153658                    2058
1013000289899876891972853643868768429576643344 -365232586204523259946104149781743467439489421 4665326151945109491433895141686203103971537555
-792959966370172313574430525096938087509165455 2858980615915271131725471989254932758686471697 -365194058228544344529990251435187084619563714
                   74                    153658                    2058                    1366
-365232586204523259946104149781743467439489421 4665326151945109491433895141686203103971537555 -34888646110598333496556928198259646436828867291
2858980615915271131725471989254932758686471697 -365194058228544344529990251435187084619563714 27310258370503808608391825805129337537716362005
                   1                    2058                    1366                    692
4665326151945109491433895141686203103971537555 -34888646110598333496556928198259646436828867291 35355178725792844445700317712428266747226021046
-365194058228544344529990251435187084619563714 27310258370503808608391825805129337537716362005 -276754524287323529529218160565645246223359257 19
                   1                    1366                    692                    674
-34888646110598333496556928198259646436828867291 35355178725792844445700317712428266747226021046 -702438248363911779422572459106879131840548883337
27310258370503808608391825805129337537716362005 -276754524287323529529218160565645246223359257 19 549857107992361615613136418616938621600522877 24
                   1                    692                    674                    18
35355178725792844445700317712428266747226021046 -702438248363911779422572459106879131840548883337 10559900356218402238795756362311617993128090938 3
-276754524287323529529218160565645246223359257 19 549857107992361615613136418616938621600522877 24 -82661163227968514514235457918258386782388213443
                   37                    674                    18                    8
-702438248363911779422572459106879131840548883337 10559900356218402238795756362311617993128090938 3 -397740695663720000629668709996598657064144853550 8
549857107992361615613136418616938621600522877 24 -82661163227968514514235457918258386782388213443 31134487502340711985880255848372541731084161851 15
                   2                    18                    8                    2
10559900356218402238795756362311617993128090938 3 -397740695663720000629668709996598657064144853550 8 80604129168365840349813317635550893212141779803 99
-82661163227968514514235457918258386782388213443 31134487502340711985880255848372541731084161851 15 -6309558663696110911690286627592766732999220583673
                   4                    8                    2                    0
-397740695663720000629668709996598657064144853550 8 80604129168365840349813317635550893212141779803 99 -362190586239835361462220141541863438554981604571 04
31134487502340711985880255848372541731084161851 15 -6309558663696110911690286627592766732999220583673 28351683405018514845349172095208321105105298519807
                   -                    2                    0                    -
80604129168365840349813317635550893212141779803 99 -362190586239835361462220141541863438554981604571 04                    -
-6309558663696110911690286627592766732999220583673 28351683405018514845349172095208321105105298519807                    -
GCD(567033668100370296906983441904166422102105970396 14, 724381172479670722924440283083726877109963209 14208) = 2
s = 80604129168365840349813317635550893212141779803 99
t = -6309558663696110911690286627592766732999220583673
```

Multiplicative Inverse

Code:

Cpp:

```cpp
#include <bits/stdc++.h>
using namespace std;

int main()
{

    int q, r1, r2, r, t1 = 0, t2 = 1, t;
    cin >> r1 >> r2;
    cout << r1 << " " << r2 << "\n";
    while (r2 > 0)
    {
        q = r1 / r2;
        r = r1 - r2 * q;
        r1 = r2;
        r2 = r;
        t = t1 - t2 * q;
        t1 = t2;
        t2 = t;
    }
    cout << r1 << " " << t1 << "\n";

    return 0;
}
```

Output:

```
d:\CNS Lab>cd "d:\CNS Lab\" && g++ multiplicative_inverse.cpp -
ultiplicative_inverse && "d:\CNS Lab\"multiplicative_inverse
4 8
4 8
4 0
```

Python:

```python
def multInv(a: int, b: int):
  r1 = a
  r2 = b
  t1 = 0
  t2 = 1
```

```
  print(f'{"q":>20} {"r1":>20} {"r2":>20} {"r":>20} {"t1":>20} {"t2":>20}
{"t":>20}')
  print(" ")

  while r2 != 0:
    q = r1 // r2
    r = r1 - q * r2
    t = t1 - q * t2

    print(f'{"%20.0d" % q} {"%20.0d" % r1} {"%20.0d" % r2} {"%20.0d" % r}
{"%20.0d" % t1} {"%20.0d" % t2} {"%20.0d" % t}')

    r1 = r2
    r2 = r
    t1 = t2
    t2 = t

  print(f'{"-":>20} {"%20.0d" % r1} {"%20.0d" % r2} {"-":>20} {"%20.0d" %
t1} {"%20.0d" % t2} {"-":>20}')

  print(f'GCD({a}, {b}) = {r1}')

  if r1 == 1:
    if t1 > 0:
      print(f'Multiplicative inverse of {b} in Z{a} = {t1}')
    else:
      print(f'Multiplicative inverse of {b} in Z{a} = {t1} or {a + t1}')
  else:
    print('Multiplicative inverse does not exist')




def main():

  a = int(input('Enter a: '))
  b = int(input('Enter b: '))
  multInv(a,b)
```

Output:

1st input:

```
main()
```

```
In [3]: runfile('D:/CNS Lab/MI.py', wdir='D:/CNS Lab')

Enter a: 1234568022

Enter b: 3322446688
                  q                r1                r2                 r                t1                t2                 t
                  0        1234568022        3322446688        1234568022                 0                 1                 0
                  2        3322446688        1234568022         853310644                 1                 0                 1
                  1        1234568022         853310644         381257378                 0                 1                -1
                  2         853310644         381257378          90795888                 1                -1                 3
                  4         381257378          90795888          18073826                -1                 3                -13
                  5          90795888          18073826            426758                 3               -13                68
                 42          18073826            426758            149990               -13                68             -2869
                  2            426758            149990            126778                68             -2869              5806
                  1            149990            126778             23212             -2869              5806             -8675
                  5            126778             23212             10718              5806             -8675             49181
                  2             23212             10718              1776             -8675             49181            -107037
                  6             10718              1776                62             49181            -107037            691403
                 28              1776                62                40            -107037            691403          -19466321
                  1                62                40                22            691403          -19466321          20157724
                  1                40                22                18          -19466321          20157724          -39624045
                  1                22                18                 4          20157724          -39624045          59781769
                  4                18                 4                 2          -39624045          59781769         -278751121
                  2                 4                 2                 0          59781769         -278751121         617284011
                  -                 2                 0                 -         -278751121         617284011                 -
GCD(1234568022, 3322446688) = 2
Multiplicative inverse does not exist
```

```
                  3            620806            155716            153658 4801007168047541910021804139641184961589753
4 -792959966370172313574430525096938087509165 2858980615915271131725471989254932758686471 69
                  1            155716            153658              2058 -792959966370172313574430525096938087509165
45 2858980615915271131725471989254932758686471 69 -365194058228544344529990251435187084619563714
                 74            153658              2058              1366 2858980615915271131725471989254932758686471 69
-365194058228544344529990251435187084619563714 2731025837050380860839182580512933753771636200
5
                  1              2058              1366               692 -365194058228544344529990251435187084619563714
2731025837050380860839182580512933753771636200
5 -276754524287323529529218160565645246223359257
19
                  1              1366               692               674 2731025837050380860839182580512933753771636200
5 -276754524287323529529218160565645246223359257
19 549857107992361615613136418616938621600522877
24
                  1               692               674                18 -276754524287323529529218160565645246223359257
19 549857107992361615613136418616938621600522877
24 -826611632279685145142354579182583867823882134
43
                 37               674                18                 8 549857107992361615613136418616938621600522877
24 -826611632279685145142354579182583867823882134
43 311344875023407119858802558483725417310841618511
5
                  2                18                 8                 2 -826611632279685145142354579182583867823882134
43 311344875023407119858802558483725417310841618511
5 -630955866369611091169028662759276673299922058367
3
                  4                 8                 2                 0 311344875023407119858802558483725417310841618511
5 -630955866369611091169028662759276673299922058367
3 283516834050185148453491720952083211051052985198
07
                  -                 2                 0                 - -630955866369611091169028662759276673299922058367
3 283516834050185148453491720952083211051052985198
07                                                  -
GCD(567033668100370296906983441904166422102105970396
14, 724381172479670722924440283083726877109963209142
08) = 2
Multiplicative inverse does not exist
```

2nd input:

```
In [1]: runfile('D:/CNS Lab/MI.py', wdir='D:/CNS Lab')

Enter a: 07512779912565014928652150296195898826835317022090

Enter b: 97569354727979857921728118865439213176360669242153
                   q                   r1                   r2                    r                   t1                   t2                    t
                   0 75127799125650149286521502961958988268353170220900 97569354727979857921728118865439213176360669242153 75127799125650149286521502961958988268353170220900
1                  0
                  12 97569354727979857921728118865439213176360669242153 75127799125650149286521502961958988268353170220900 74159957771996787779023153110884272543368649770730
0                  1
                   1 75127799125650149286521502961958988268353170220900 74159957771996787779023153110884272543368649770730 96784135365336150749834985107471572498452045017
```

```
-12581359651207264965131327317302879238440718670
                   9                 5470                  589                  169 10709273062048234171906863389324683928832664000 -12581359651207264965131327317302879238440718670
12394150992291361885808880924505059707479913203
                   3                  589                  169                   82 -12581359651207264965131327317302879238440718670 12394150992291361885808880924505059707479913203
-38440588941994812153939775505245467046283811476
                   2                  169                   82                    5 12394150992291361885808880924505059707479913203 -38440588941994812153939775505245467046283811476
89275328876280986193688431934995993800047536155
                  16                   82                    5                    2 -38440588941994812153939775505245467046283811476 89275328876280986193688431934995993800047536155
-14668458509624905912529546864651813678470443899560
                   2                    5                    2                    1 89275328876280986193688431934995993800047536155 -14668458509624905912529546864651813678470443899560
30229670308012621686995978048653587294941363160670
                   2                    2                    1                    0 -14668458509624905912529546864651813678470443899560 30229670308012621686995978048653587294941363160670
-75127799125650149286521502961958988268353170220900
                   -                    1                    0                    - 30229670308012621686995978048653587294941363160670 -75127799125650149286521502961958988268353170220900
GCD(75127799125650149286521502961958988268353170220900, 97569354727979857921728118865439213176360669242153) = 1
Multiplicative inverse of 97569354727979857921728118865439213176360669242153 in Z75127799125650149286521502961958988268353170220900 = 30229670308012621686995978048653587294941363160670
```