Question 1

In [21]:
```
from functionLibrary import gaussj

matrix = open("q1-matrix.txt", "r+")

M = []
N = []

for row in matrix:
    e1 = row.split()
    fe1 = []
    for i in range(len(e1)-1):
        fe1.append(float(e1[i]))
    M.append(fe1)
    e2 = row.split()
    N.append(float(e2[len(e1)-1]))

print("The solution of the given system of equations is:")

gaussj(M,N)
```

```
The solution of the given system of equations is:
[2.000000000000001, -1.3322676295501878e-15, 5.999999999999999, 5.0]
```

Question 2

In [22]:
```
from functionLibrary import gaussj

matrix = open("q2-matrix.txt", "r+")

M = []
N = []

for row in matrix:
    e1 = row.split()
    fe1 = []
    for i in range(len(e1)-1):
        fe1.append(float(e1[i]))
    M.append(fe1)
    e2 = row.split()
    N.append(float(e2[len(e1)-1]))

print("The solution of the given system of equations is:")

gaussj(M,N)
```

```
The solution of the given system of equations is:
[1.0, -2.0, -1.0]
```

Question 3

In [24]:
```
from functionLibrary import inverse
from functionLibrary import matrixProduct

matrix = open("q3-matrix.txt", "r+")

M = []
for row in matrix:
    e1 = row.split()
    fe1 = []
    for i in range(len(e1)):
        fe1.append(float(e1[i]))
    M.append(fe1)

print("The inverse of the given matrix is: ")

inverse(M)


#Verification
#the construction of R and S was asked to limit the matrix entries to two decimal spaces.
#first constructing the matrix with string entries and then converting that into floats is the only viable option in Python
#re-constructing the matrix with values upto two decimal places as strings
n = len(M)
R = []
for i in range(len(M)):
    r = []
    for j in range(n, len(M[0])):
        r.append("{:.2f}".format(M[i][j]))
    R.append(r)

#defining matrix to have numbers rounded off to 2 decimal places as floats
S = []
for i in range(len(R)):
    s = []
    for j in range(len(R[0])):
        s.append(float(R[i][j]))
    S.append(s)


matrix = open("q3-matrix.txt", "r+")

Q = []
for row in matrix:
    e1 = row.split()
    fe1 = []
    for i in range(len(e1)):
        fe1.append(float(e1[i]))
    Q.append(fe1)



matrixProduct(Q, S)

print("Clearly, above matrix is an identity matrix.")
```

```
The inverse of the given matrix is:
-0.33 0.33 0.33
-0.17 0.17 0.67
1.33 -0.33 -1.33
Multiplying the two matrices:
0.99 0.010000000000000009 0.010000000000000009
0.010000000000000009 0.99 -0.010000000000000009
-0.010000000000000009 0.010000000000000009 1.01
Clearly, above matrix is an identity matrix.
```

Question 4

In [25]:
```
from functionLibrary import determinant

matrix = open("q4-matrix.txt", "r+")

M = []
for row in matrix:
    e1 = row.split()
    fe1 = []
    for i in range(len(e1)):
        fe1.append(float(e1[i]))
    M.append(fe1)

print("The determinant of the given matrix is: ")

determinant(M)
```

```
The determinant of the given matrix is:
65.0
```