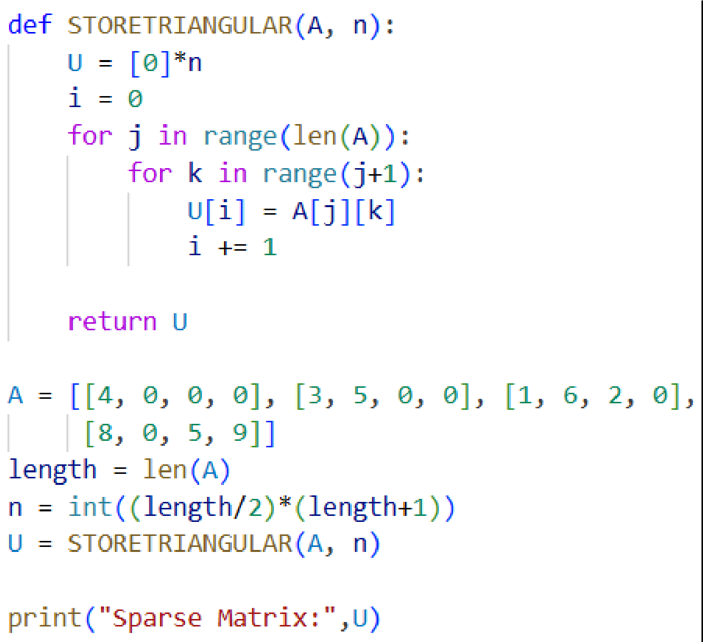
USMAN RASHEED SIDDIQUI CS-24038
NED UNIVERSITY OF ENGINEERING AND TECHNOLOGY DATA STRUCTURES AND ALGORITM

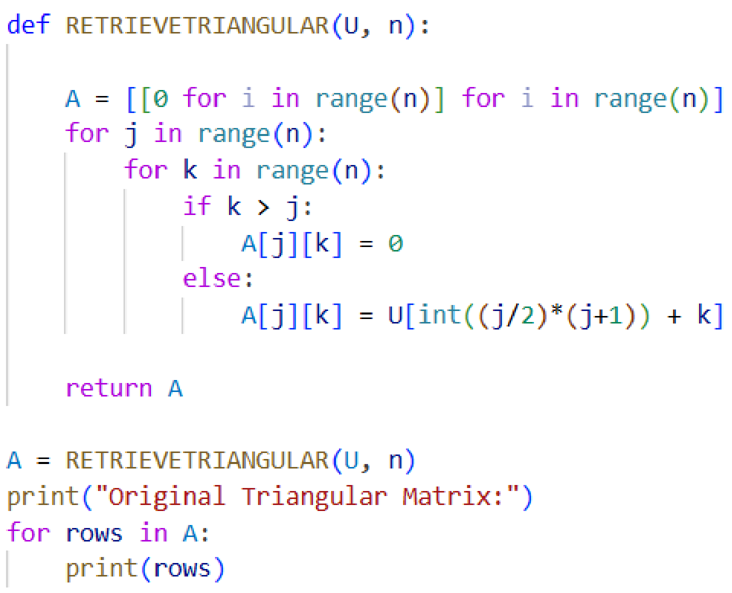
Lab Session 5

Ex A:

CODE

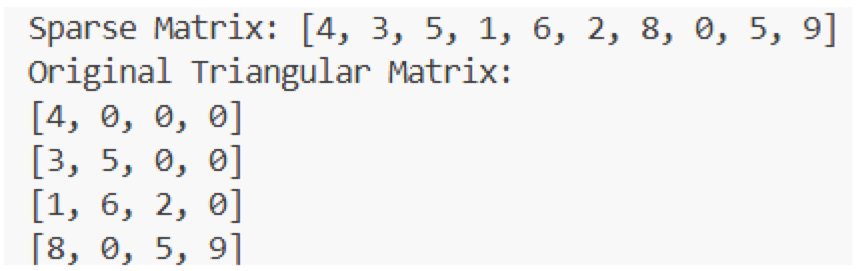


```python
def STORETRIANGULAR(A, n):
    U = [0]*n
    i = 0
    for j in range(len(A)):
        for k in range(j+1):
            U[i] = A[j][k]
            i += 1

    return U

A = [[4, 0, 0, 0], [3, 5, 0, 0], [1, 6, 2, 0],
     [8, 0, 5, 9]]
length = len(A)
n = int((length/2)*(length+1))
U = STORETRIANGULAR(A, n)

print("Sparse Matrix:",U)
```



```python
def RETRIEVETRIANGULAR(U, n):

    A = [[0 for i in range(n)] for i in range(n)]
    for j in range(n):
        for k in range(n):
            if k > j:
                A[j][k] = 0
            else:
                A[j][k] = U[int((j/2)*(j+1)) + k]

    return A

A = RETRIEVETRIANGULAR(U, n)
print("Original Triangular Matrix:")
for rows in A:
    print(rows)
```

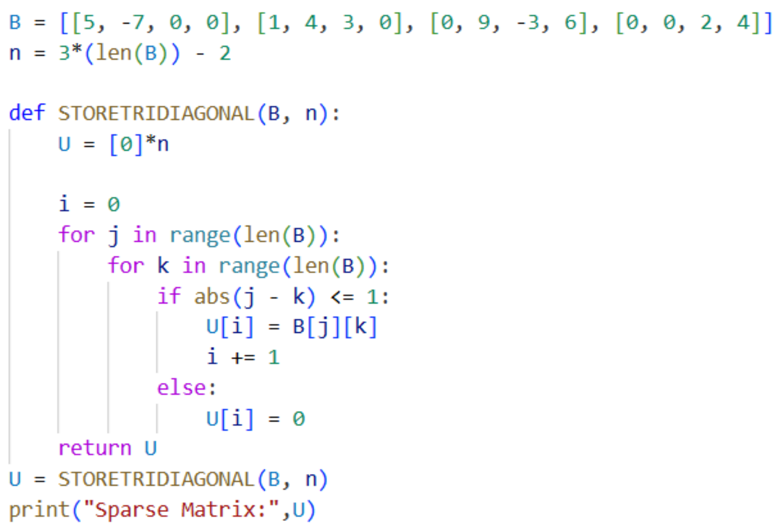
OUTPUT



```
Sparse Matrix: [4, 3, 5, 1, 6, 2, 8, 0, 5, 9]
Original Triangular Matrix:
[4, 0, 0, 0]
[3, 5, 0, 0]
[1, 6, 2, 0]
[8, 0, 5, 9]
```

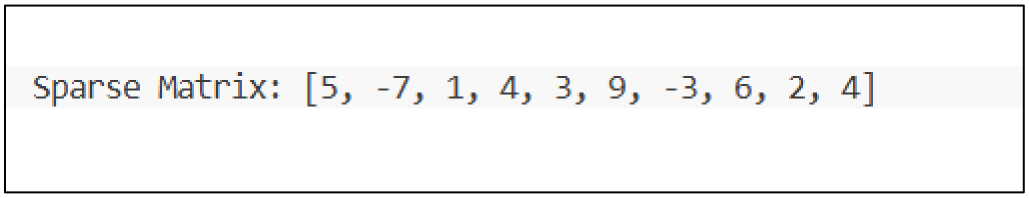
Ex D:

CODE



```python
B = [[5, -7, 0, 0], [1, 4, 3, 0], [0, 9, -3, 6], [0, 0, 2, 4]]
n = 3*(len(B)) - 2

def STORETRIDIAGONAL(B, n):
    U = [0]*n

    i = 0
    for j in range(len(B)):
        for k in range(len(B)):
            if abs(j - k) <= 1:
                U[i] = B[j][k]
                i += 1
            else:
                U[i] = 0
    return U
U = STORETRIDIAGONAL(B, n)
print("Sparse Matrix:",U)
```

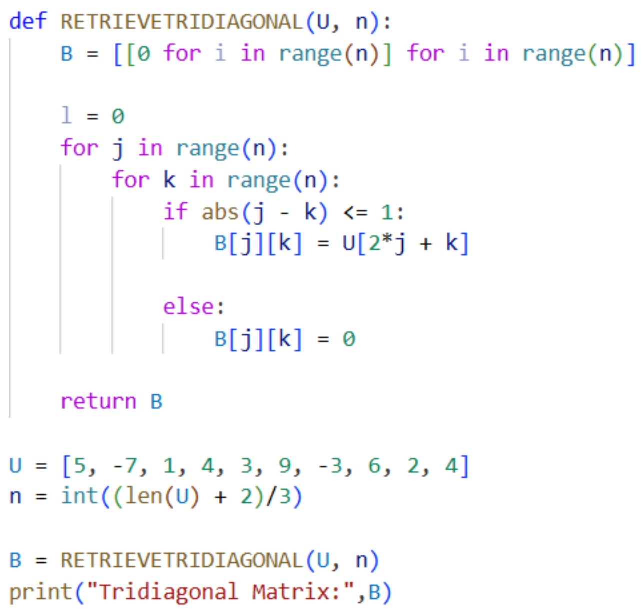
OUTPUT



```
Sparse Matrix: [5, -7, 1, 4, 3, 9, -3, 6, 2, 4]
```

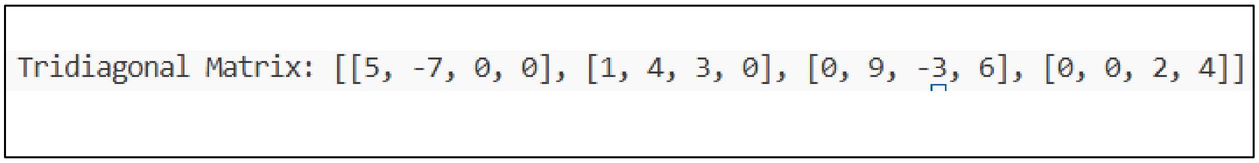
Ex E:

CODE



```python
def RETRIEVETRIDIAGONAL(U, n):
    B = [[0 for i in range(n)] for i in range(n)]

    l = 0
    for j in range(n):
        for k in range(n):
            if abs(j - k) <= 1:
                B[j][k] = U[2*j + k]

            else:
                B[j][k] = 0

    return B

U = [5, -7, 1, 4, 3, 9, -3, 6, 2, 4]
n = int((len(U) + 2)/3)

B = RETRIEVETRIDIAGONAL(U, n)
print("Tridiagonal Matrix:",B)
```

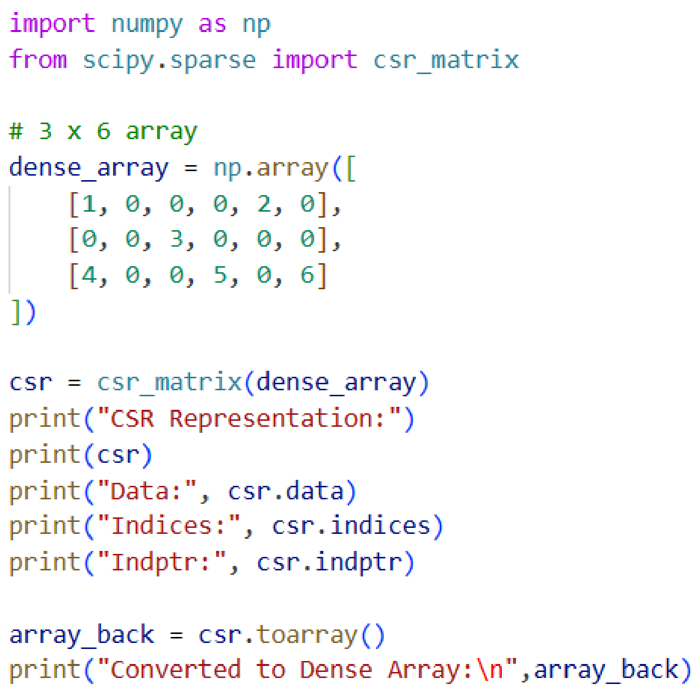
OUTPUT



```
Tridiagonal Matrix: [[5, -7, 0, 0], [1, 4, 3, 0], [0, 9, -3, 6], [0, 0, 2, 4]]
```

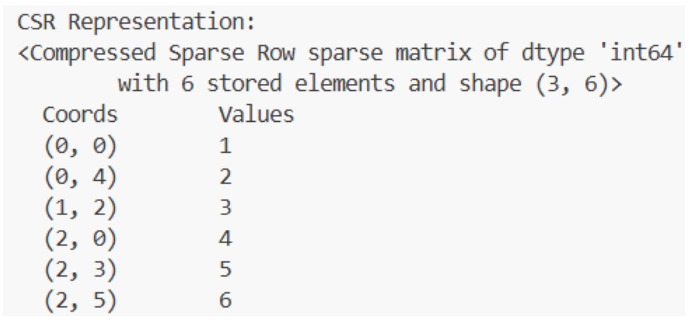
Ex F:

CODE

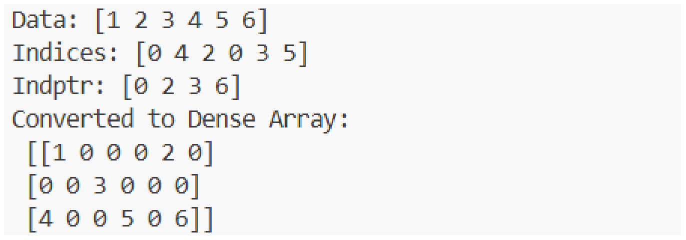


```python
import numpy as np
from scipy.sparse import csr_matrix

# 3 x 6 array
dense_array = np.array([
    [1, 0, 0, 0, 2, 0],
    [0, 0, 3, 0, 0, 0],
    [4, 0, 0, 5, 0, 6]
])

csr = csr_matrix(dense_array)
print("CSR Representation:")
print(csr)
print("Data:", csr.data)
print("Indices:", csr.indices)
print("Indptr:", csr.indptr)

array_back = csr.toarray()
print("Converted to Dense Array:\n",array_back)
```

OUTPUT



```
CSR Representation:
<Compressed Sparse Row sparse matrix of dtype 'int64'
        with 6 stored elements and shape (3, 6)>
  Coords        Values
  (0, 0)        1
  (0, 4)        2
  (1, 2)        3
  (2, 0)        4
  (2, 3)        5
  (2, 5)        6
```



```
Data: [1 2 3 4 5 6]
Indices: [0 4 2 0 3 5]
Indptr: [0 2 3 6]
Converted to Dense Array:
 [[1 0 0 0 2 0]
 [0 0 3 0 0 0]
 [4 0 0 5 0 6]]
```