# Part a

```python
#A5.1
def storeTriangular(A):
    n = len(A)
    U = [0] * (n * (n + 1) // 2)
    i = 0
    for j in range(n):
        for k in range(j + 1):
            U[i] = A[j][k]
            i += 1
    return U
# A5.2
def retrieveTriangular(U, n):
    A = [[0]*n for _ in range(n)]
    for j in range(n):
        for k in range(n):
            if k <= j:
             idx = (j*(j+1))//2 + k
             A[j][k] = U[idx]
            else:
                A[j][k] = 0
    return A
A = [
    [1, 0, 0],
[4, 5, 0],[7, 8, 9]]
U = storeTriangular(A)
print("Stored U:", U)
n = len(A)
retrieved_A = retrieveTriangular(U, n)
print("Retrieved A:")
for row in retrieved_A:
    print(row)
```

## OUTPUT

```
PS C:\Users\NB\Desktop\CisTh&Lab\Cis-2024-Muzammil\DSATsks>
Cis-2024-Muzammil\DSATsks\lab5\actual\tempCodeRunnerFile.py'
Stored U: [1, 4, 5, 7, 8, 9]
Retrieved A:
[1, 0, 0]
[4, 5, 0]
[7, 8, 9]
PS C:\Users\NB\Desktop\CisTh&Lab\Cis-2024-Muzammil\DSATsks>
```

# ***Part d***

```
def store_tridiagonal(B):
    n = len(B)
    size_U = 3 * n - 2
    U = [0] * size_U
    for j in range(n):
        for k in range(n):
            if j == k:  # main diagonal
                L = j
                U[L] = B[j][k]
            elif k == j + 1:  # upper diagonal
                L = n + j
                U[L] = B[j][k]
            elif k == j - 1:  # lower diagonal
                L = 2 * n - 2 + (j - 1)
                U[L] = B[j][k]
    return U
B = [
    [5, -7, 0, 0],
    [1, 4, 3, 0],
    [0, 9, -3, 6],[0, 0, 2, 4]]

U = store_tridiagonal(B)
print("U =", U)
```

# Output

```
PS C:\Users\NB\Desktop\CisTh&Lab\Cis-2024-Muzammil\DSATsks>
Cis-2024-Muzammil\DSATsks\lab5\actual\tempCodeRunnerFile.py"
U = [5, -7, 1, 4, 3, 9, -3, 6, 2, 4]
PS C:\Users\NB\Desktop\CisTh&Lab\Cis-2024-Muzammil\DSATsks>
```

# Part e

task3.py | task1.py | partd.py | partF.py | partE.py

lab5 > actual > partE.py > ...

```python
def retrievTridiagonal(U, n):
    B = [[0] * n for _ in range(n)]
    for j in range(n):
        B[j][j] = U[j] # main diagonal
        if j < n - 1: # upper diagonal
            B[j][j + 1] = U[n + j]
        if j > 0: # lower diagonal
            B[j][j - 1] = U[2 * n - 2 + (j - 1)]

    return B
U = [5, 4, -3, 4, -7, 3, 9, 6, 1, 2]
B = retrievTridiagonal(U, 4)
print("Retrieved B:")
for row in B:
    print(row)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\NB\Desktop\CisTh&Lab\Cis-2024-Muzammil\DSATsks> python -u "c:\Users\NB\D
Cis-2024-Muzammil\DSATsks\lab5\actual\partE.py"
Retrieved B:
[5, -7, 0, 0]
[9, 4, 3, 0]
[0, 6, -3, 9]
[0, 0, 1, 4]
PS C:\Users\NB\Desktop\CisTh&Lab\Cis-2024-Muzammil\DSATsks>
```

---

# Part f

task3.py | task1.py | partd.py | partF.py | partE.py

lab5 > actual > partF.py > ...

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

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

sparse_csr = csr_matrix(dense_matrix)
print("Non-zero values:", sparse_csr.data)

print("Dense form:\n", sparse_csr.toarray())

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\NB\Desktop\CisTh&Lab\Cis-2024-Muzammil\DSATsks> python -u "c:\Users\N
Cis-2024-Muzammil\DSATsks\lab5\actual\partF.py"
Non-zero values: [1 2 3 4 5 6]
Dense form:
 [[1 0 0 0 2 0]
 [0 0 3 0 0 4]
 [5 0 0 6 0 0]]
PS C:\Users\NB\Desktop\CisTh&Lab\Cis-2024-Muzammil\DSATsks>
```