Name – Harsh Zanwar

Roll no. – 77

PRN – 12110333

Class – AI & DS – C

Subject – OS Lab Assignment 7

Problem Statement : Implement Bankers algorithm for deadlock avoidance and find out a safe sequence for processes.

Code :

```
import numpy as np

class BankersAlgorithm:
    def __init__(self, allocation, max_resources, available):
        self.allocation = np.array(allocation)
        self.max_resources = np.array(max_resources)
        self.available = np.array(available)
        self.need = self.max_resources - self.allocation
        self.num_processes, self.num_resources = self.allocation.shape

    def is_safe_state(self, sequence):
        work = self.available.copy()
        finish = np.zeros(self.num_processes, dtype=bool)

        for i in sequence:
            if not finish[i] and np.all(self.need[i] <= work):
                work += self.allocation[i]
                finish[i] = True

        return np.all(finish)

    def find_safe_sequence(self):
        work = self.available.copy()
        finish = np.zeros(self.num_processes, dtype=bool)
        safe_sequence = []

        while len(safe_sequence) < self.num_processes:
            found = False
```

```
            for i in range(self.num_processes):
                if not finish[i] and np.all(self.need[i] <= work):
                    work += self.allocation[i]
                    finish[i] = True
                    safe_sequence.append(i)
                    found = True
                    break
            if not found:

                return None

        return safe_sequence

def get_matrix(prompt, rows, cols):
    matrix = []
    print(prompt)
    for i in range(rows):
        row = list(map(int, input().split()))
        if len(row) != cols:
            print("Invalid input. Please enter exactly", cols, "values.")
            return get_matrix(prompt, rows, cols)
        matrix.append(row)
    return matrix


allocation = get_matrix("Enter the allocation matrix (one row per process):", 5,
3)


max_resources = get_matrix("Enter the maximum resources matrix (one row per
process):", 5, 3)


available = list(map(int, input("Enter the available resources: ").split()))

banker = BankersAlgorithm(allocation, max_resources, available)
safe_sequence = banker.find_safe_sequence()

if safe_sequence is not None:
    print("Safe Sequence:", safe_sequence)
else:
    print("No safe sequence found.")
```

Output :

```
PS C:\Users\VINAY\Desktop\OS Lab> & C:/Users/VINAY/anaconda3/python.exe "c:/Users/VINAY/D
y"
Enter the allocation matrix (one row per process):
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
Enter the maximum resources matrix (one row per process):
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
Enter the available resources: 3 3 2
Safe Sequence: [1, 3, 0, 2, 4]
PS C:\Users\VINAY\Desktop\OS Lab> & C:/Users/VINAY/anaconda3/python.exe "c:/Users/VINAY/D
```

```
PS C:\Users\VINAY\Desktop\OS Lab> & C:/Users/VINAY/anaconda3/python.exe "c:/Users/VINAY
y"
Enter the allocation matrix (one row per process):
3 3 3
3 2 1
1 2 3
1 4 5
1 2 5
Enter the maximum resources matrix (one row per process):
3 4 3
2 1 4
1 3 5
1 3 6
3 4 6
Enter the available resources: 1 2 4
Safe Sequence: [0, 1, 2, 3, 4]
PS C:\Users\VINAY\Desktop\OS Lab>
```