|  |  |  |  |
| --- | --- | --- | --- |
| **Fr. Conceicao Rodrigues College of Engineering**  **Department of Computer Engineering** | | | |
| **Student’s Roll No** |  | **Students Name** |  |
| **Date of Performance** |  | **SE Computer – Div** | **A / B** |

**Aim:** To study Deadlock detection and Avoidance strategies

**Lab Outcome:**

**CSL403.3:** Understand and apply the concepts of synchronization and deadlocks

**Pre-requirement: Python Programming.**

**Problem Statements:**

WAP for the following.

Inputs: Number of processes, No of Resources, Instances of each resources, Number of resources held by each process , Number of resources needed by each process/Maximum number of resources needed by each process.

Write a menu driven program.

1) Detect if a deadlock exists. Also show the processes involved in deadlock

2) Check if the deadlock can be avoided (using bankers algo.). If yes, give the safe state sequence.

**References:**

<https://www.javatpoint.com/os-resource-allocation-graph>

<https://www.javatpoint.com/os-deadlock-avoidance>

|  |  |  |  |
| --- | --- | --- | --- |
| **On time Submission(2)** | **Knowledge of Topic(4)** | **Implementation and Demonstraion(4)** | **Total (10)** |
|  |  |  |  |
| **Signature of Faculty** |  | **Date of Submission** |  |

Banker’s Algorithm:

import numpy as np

process = [1, 2, 3, 4, 5]

allocation\_matrix = np.array([[0,1,0],[2,0,0],[3,0,2],[2,1,1],[0,0,2]])

max\_matrix = np.array([[8,5,3],[3,2,2],[11,0,2],[2,2,2],[4,3,3]])

available\_matrix = np.array([3,3,2])

need = np.array(max\_matrix-allocation\_matrix)

safe\_sequence = []

complete = [0]\*5

n = len(process)

count=0

while(count<n):

    found = False

    for i in range(n):

        if np.all(np.greater(need[i,:],available\_matrix)):

            complete[i] = 0

        elif (np.all(np.less\_equal(need[i,:],available\_matrix)) and complete[i]==0):

            safe\_sequence.append(process[i])

            print("process is being executed")

            print("process ",i+1," needs -> ",need[i,:]," resources")

            print("process ",i+1," using -> ",max\_matrix[i,:]," resources")

            print("available resources -> ",available\_matrix-need[i,:])

            available\_matrix += allocation\_matrix[i,:]

            allocation\_matrix[i,:] =[0,0,0]

            print('process ',i+1, ' completed\nresources will be released')

            print("available resources -> ",available\_matrix)

            print()

            complete[i] = 1

            count +=1

            found = True

    if not found:

        print('system is not in safe state')

        break

if (all(x == 1 for x in complete)):

    print('system is in a safe state')

    print(safe\_sequence)