|  |  |
| --- | --- |
| Name | Manish Shashikant Jadhav |
| UID no. | 2023301005 |

|  |
| --- |
| Experiment 8 |

|  |  |
| --- | --- |
| **Aim** | Considering a system with five processes P0 through P4 and three resources of type A, B, C. Resource type A has 10 instances, B has 5 instances and type C has 7 instances. Suppose at time t0 following snapshot of the system has been taken:    Question1. What will be the content of the Need matrix?  Question2. Is the system in a safe state? If Yes, then what is the safe sequence?  Question3. What will happen if process P1 requests one additional instance of resource type A and two instances of resource type C?  Implement using Any programming language. |
| **Code:** | *# 5  processes: P0, P1, P2, P3, P4*  n = 5  *# Number of processes: P0, P1, P2, P3, P4*  m = 3  *# Number of resources: A, B, C*  *# ALLOCATION COLUMN*  alloc = [[0, 1, 0], [2, 0, 0],           [3, 0, 2], [2, 1, 1], [0, 0, 2]]  *# MAXIMUM COLUMN*  max = [[7, 5, 3], [3, 2, 2],         [9, 0, 2], [2, 2, 2], [4, 3, 3]]  *# AVAILABLE COLUMN*  avail = [3, 3, 2]  *# table display for output which includes process, allocation, max, available based on input values provided.*  print("Process      Allocation           Max         Available")  *for* i *in* range(n):      print(f"P{i}            {alloc[i]}        {max[i]}      {avail}")  f = [0] \* n  ans = [0] \* n  ind = 0  *for* k *in* range(n):      f[k] = 0  need = [[0 *for* i *in* range(m)] *for* i *in* range(n)]  *for* i *in* range(n):  *for* j *in* range(m):          need[i][j] = max[i][j] - alloc[i][j]  print("\nNeed Matrix calculated = (Need(i, j) = Max(i, j) - Allocation(i, j)):")  *for* i *in* range(n):      print(need[i])  *for* k *in* range(5):  *for* i *in* range(n):  *if* f[i] == 0:              flag = 0  *for* j *in* range(m):  *if* need[i][j] > avail[j]:                      flag = 1  *break*  *if* flag == 0:                  ans[ind] = i                  ind += 1  *for* y *in* range(m):                      avail[y] += alloc[i][y]                  f[i] = 1  *break*        print("\nAfter allocation of P" + str(ans[ind - 1]) + " Available: " + str(avail))  print("\nThe safe sequence for the program is as follows:")  *for* i *in* range(n - 1):      print(" P", ans[i], " ->", *sep*="", *end*="")  print(" P", ans[n - 1], *sep*="") |
| **Output**: |  |
| **Questions** |  |
| **Conclusion** | Hence, by completing this experiment I came to know about Banker’s Algorithm. |