BANKER\_ALGORITHM:

1.Write a Python program to implement the Banker’s Algorithm for the above used matrices.

Banker.sh:

|  |
| --- |
| P = 5 # Number of processes  R = 3 # Number of resources  def calculateNeed(need, maxm, allot):  for i in range(P):  for j in range(R):  need[i][j] = maxm[i][j] - allot[i][j]  def isSafe(processes, avail, maxm, allot):  need = [[0] \* R for \_ in range(P)]  calculateNeed(need, maxm, allot)  finish = [0] \* P  safeSeq = [0] \* P  work = avail[:]  count = 0  while count < P:  found = False  for p in range(P):  if finish[p] == 0:  if all(need[p][j] <= work[j] for j in range(R)):  for j in range(R):  work[j] += allot[p][j]  safeSeq[count] = p  count += 1  finish[p] = 1  found = True  if not found:  print("System is not in a safe state")  return False  print("System is in a safe state.")  print("Safe sequence is:", ' -> '.join(f'P{p}' for p in safeSeq))  return True  if \_\_name\_\_ == "\_\_main\_\_":  processes = [0, 1, 2, 3, 4]  avail = [3, 3, 2]  maxm = [  [7, 5, 3], # P0  [3, 2, 2], # P1  [9, 0, 2], # P2  [2, 2, 2], # P3  [4, 3, 3] # P4]  allot = [  [0, 1, 0], # P0  [2, 0, 0], # P1  [3, 0, 2], # P2    [2, 1, 1], # P3  [0, 0, 2] # P4 ]  isSafe(processes, avail, maxm, allot) |

OUTPUT:

