| | |
|---|---|
| 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:

| Process | Allocation | Max | Available |
|---|---|---|---|
| | A B C | A B C | A B C |
| $P_0$ | 0 1 0 | 7 5 3 | 3 3 2 |
| $P_1$ | 2 0 0 | 3 2 2 | |
| $P_2$ | 3 0 2 | 9 0 2 | |
| $P_3$ | 2 1 1 | 2 2 2 | |
| $P_4$ | 0 0 2 | 4 3 3 | |

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**:

```
PS D:\Manish\SPIT> & C:/Users/manis/AppData/Local/Programs/Python/Python311/python.exe "d:/Manish/SPIT/4th SEM/OS/Experiments/Exp8/bankers.py"
 Process      Allocation          Max         Available
 P0           [0, 1, 0]       [7, 5, 3]     [3, 3, 2]
 P1           [2, 0, 0]       [3, 2, 2]     [3, 3, 2]
 P2           [3, 0, 2]       [9, 0, 2]     [3, 3, 2]
 P3           [2, 1, 1]       [2, 2, 2]     [3, 3, 2]
 P4           [0, 0, 2]       [4, 3, 3]     [3, 3, 2]

 Need Matrix calculated = (Need(i, j) = Max(i, j) - Allocation(i, j)):
 [7, 4, 3]
 [1, 2, 2]
 [6, 0, 0]
 [0, 1, 1]
 [4, 3, 1]

 After allocation of P1 Available: [5, 3, 2]

 After allocation of P3 Available: [7, 4, 3]

 After allocation of P0 Available: [7, 5, 3]

 After allocation of P2 Available: [10, 5, 5]

 After allocation of P4 Available: [10, 5, 7]

 The safe sequence for the program is as follows:
  P1 -> P3 -> P0 -> P2 -> P4
PS D:\Manish\SPIT>
```

**Questions**

2023301005

OS Experiment No. 8.

Page No.: Date: YOUVA

Q.1) What will be the content of need matrix?

⇒
$$\begin{bmatrix} 7,4,3 \\ 1,2,2 \\ 6,0,0 \\ 0,1,1 \\ 4,3,1 \end{bmatrix}$$

Q.2) Is the system in safe state? If yes, then what is the sequence?

⇒ The system is in safe state.

$P_1 \rightarrow P_3 \rightarrow P_4 \rightarrow P_0 \rightarrow P_2$

Q.3) What will happen if process $P_1$ requests one additional instance of resource type A and two instances of resource type C?

⇒

| A | B | C |
|---|---|---|
| 1 | 0 | 2 |

$1,0,2 < 1,2,2$

$Request_1 \quad Need_1$

② $1,0,2 < 3,3,2$

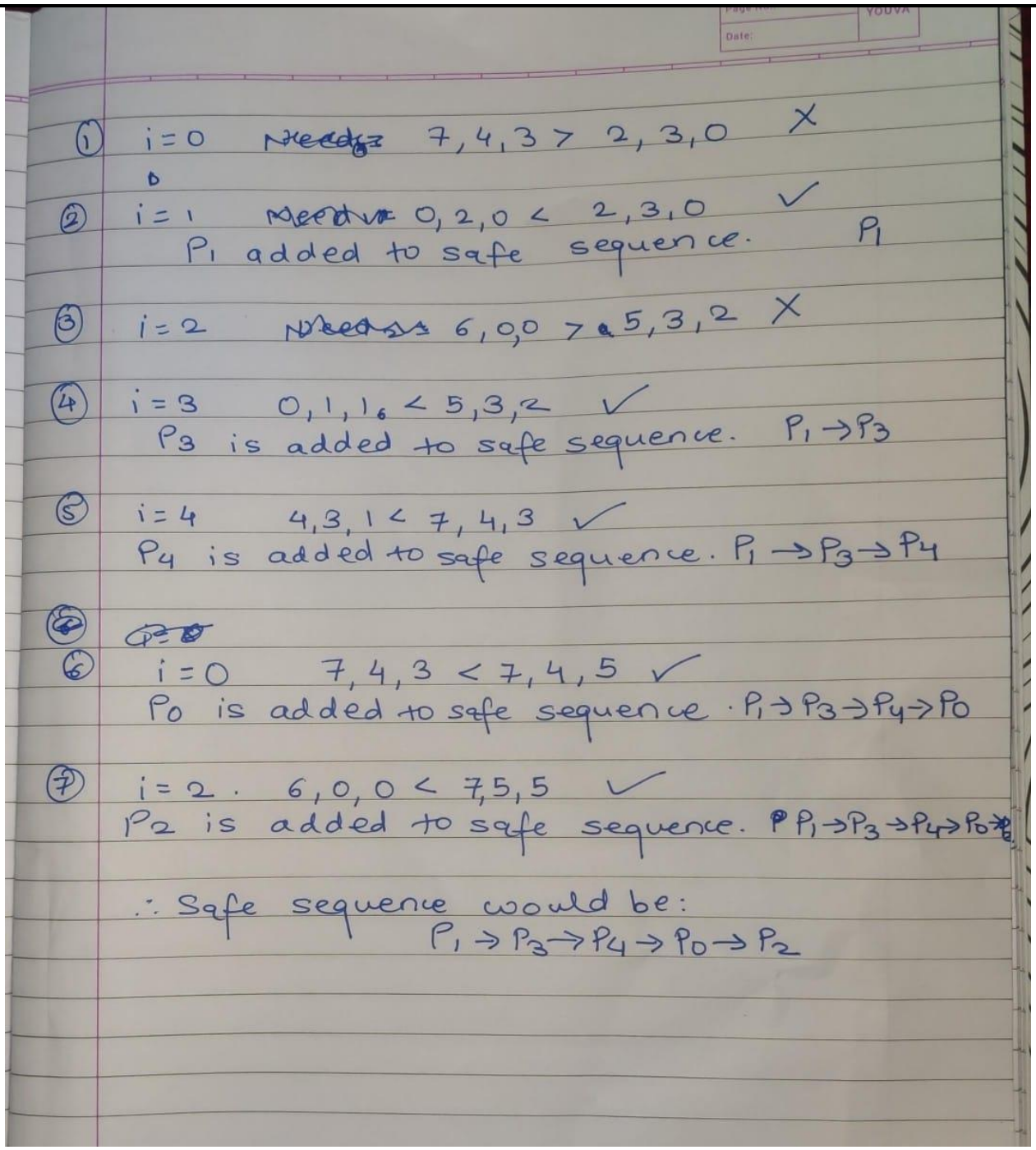
$Request_1 \quad Available$

Available = Available − $Request_1$

Allocation = $Allocation_1$ + $Request_1$

$Need_1$ = $Need_1$ − Request

| Process | Allocation | | | Need | | | Available | | |
|---|---|---|---|---|---|---|---|---|---|
| | A | B | C | A | B | C | A | B | C |
| $P_0$ | 0 | 1 | 0 | 7 | 4 | 3 | 2 | 3 | 0 |
| $P_1$ | 3 | 0 | 2 | 0 | 2 | 0 | | | |
| $P_2$ | 3 | 0 | 2 | 6 | 0 | 0 | | | |
| $P_3$ | 2 | 1 | 1 | 0 | 1 | 1 | | | |
| $P_4$ | 0 | 0 | 2 | 4 | 3 | 1 | | | |




| | |
|---|---|
| **Conclusion** | Hence, by completing this experiment I came to know about Banker's Algorithm. |