Dated: XX/XX/2021

Experiment No. 06

Objective: Implementation of Bankers Algorithm

Program:

```
def calNeed(max,allocation):
    need=[]
    for i in range(5):
        l=[]
        for j in range(3):
            l.append(max[i][j]-allocation[i][j])
        need.append(l)
    return need

def safe_seq(availabe, max, allocatin, need):
    n=len(allocatin)
    work=availabe.copy()
    finish=[False]*n
    ans=[]
    k=0
    while k<len(allocatin):
        for i in range(len(allocatin)):
            if finish[i]==False:
                if need[i][0]<=work[0] and need[i][1]<=work[1] and need[i][2]<=work[2]:
                    ans.append(f"P{i}")
                    for j in range(3):
                        allocatin[i][j]+=need[i][j]
                        work[j] -= need[i][j]
                        need[i][j]=max[i][j]-allocatin[i][j]
                    for j in range(3):
                        work[j] += allocatin[i][j]
                    finish[i]=True
        k+=1
    return ans

if __name__ == "__main__":
    availabe=[3,3,2]
    max=[[7,5,3],[3,2,2],[9,0,2],[2,2,2],[4,3,3]]
    allocation=[[0,1,0],[2,0,0],[3,0,2],[2,1,1],[0,0,2]]
    need = calNeed(max, allocation)

    while True:
        p = int(input("ENTER PROCESS ID  :: "))
        req = list(map(int, input("ENTER REQUEST  :: ").split()))
        flag=False
        if req[0]<=need[p][0] and req[1]<=need[p][1] and req[2]<=need[p][2]: #step 1
            if req[0]<=availabe[0] and req[1]<=availabe[1] and req[2]<=availabe[2]: #step 2
                for i in range(len(req)):
                    availabe[i] -=req[i]
                    allocation[p][i]+=req[i]
                    need[p][i] -=req[i]
                safeSeq=safe_seq(availabe, max, allocation, need)
                flag=True
        if len(        )==5:
            print("SAFE SEQUENCE ",*safeSeq)
        else:
            print("SAFE SEQUENCE NOT OBTAINED")
            for i in range(len(req)):
                availabe[i] += req[i]
                allocation[p][i] -= req[i]
                need[p][i] += req[i]
        if flag==False:
            print("SAFE SEQUENCE NOT OBTAINED")
        con=input("WANT TO CONTINUE TYPE Y/N")
        if con=="n" or con=="N":
            break
```

Input/Output:

```
 SAFE SEQUENCE P1 P3 P4 P0 P2
```

```
ENTER PROCESS ID  :: 4
ENTER REQUEST  :: 3 3 0
SAFE SEQUENCE NOT OBTAINED
WANT TO CONTINUE TYPE Y/Ny
ENTER PROCESS ID  :: 0
ENTER REQUEST  :: 0 2 0
SAFE SEQUENCE  P3 P1 P2 P0 P4
WANT TO CONTINUE TYPE Y/Nn
```