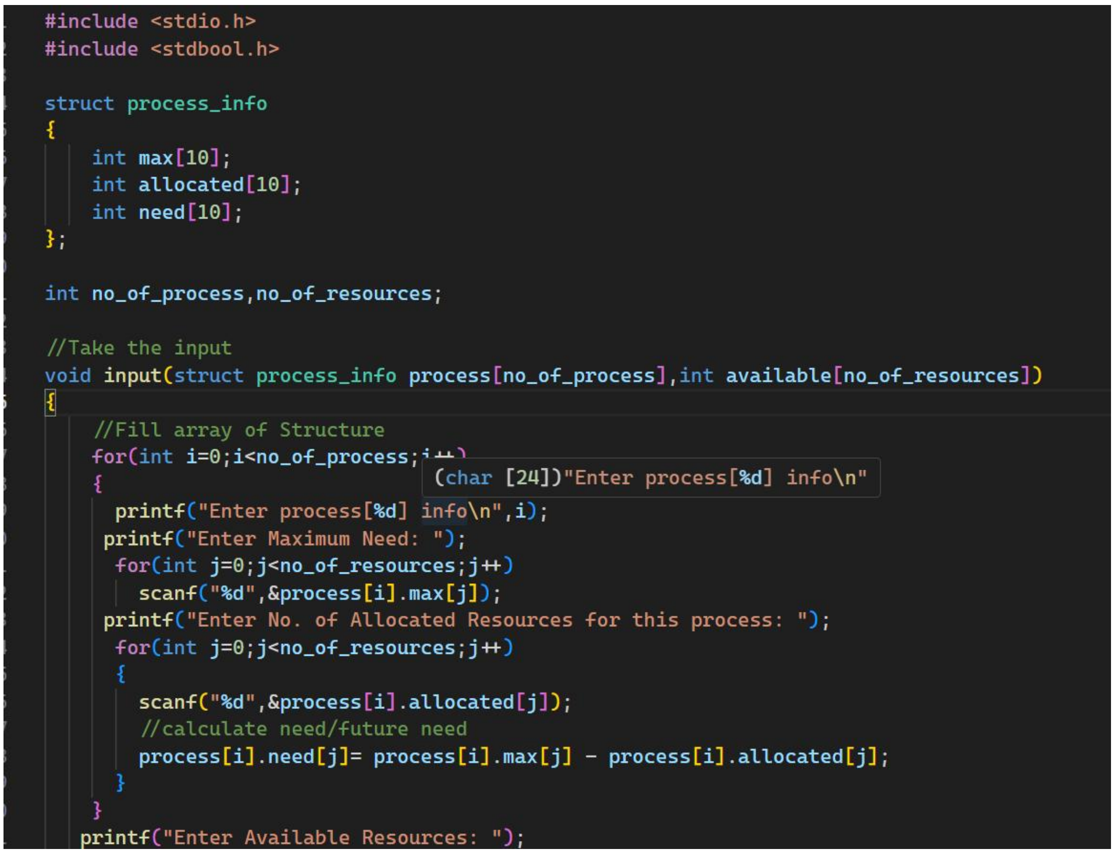
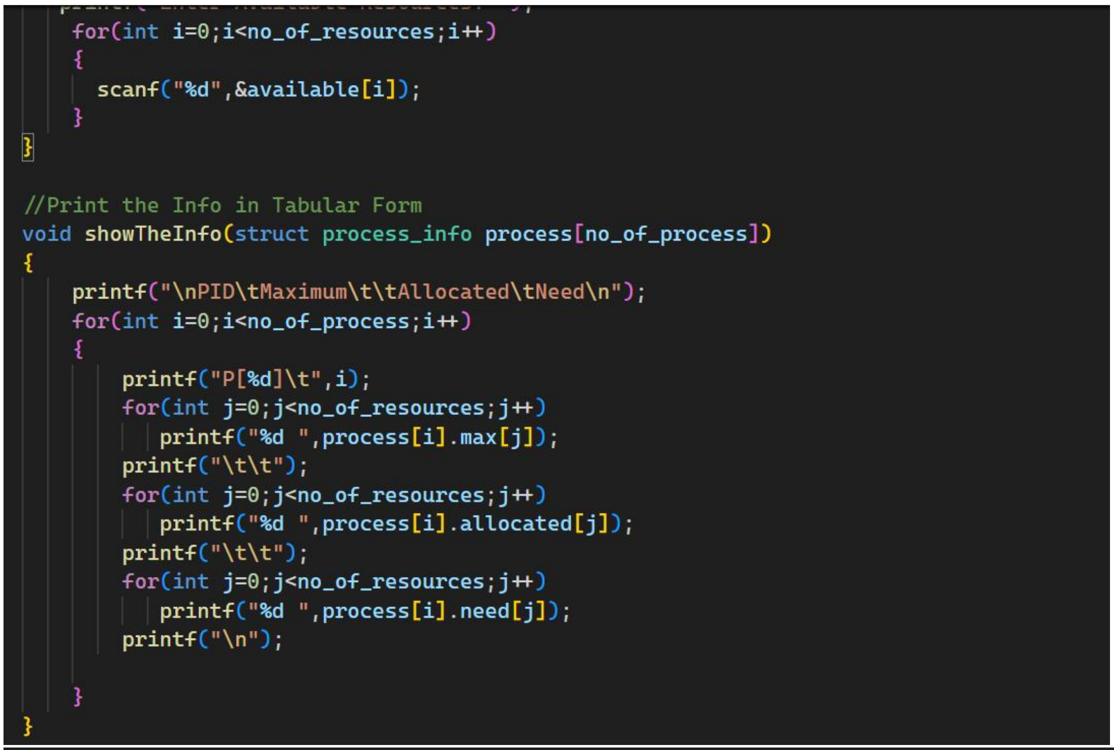
Name : Kunal Sachin Kharat
Roll no : 33
Class : CSAIML-A

Q.) Implement Bankers Algorithm

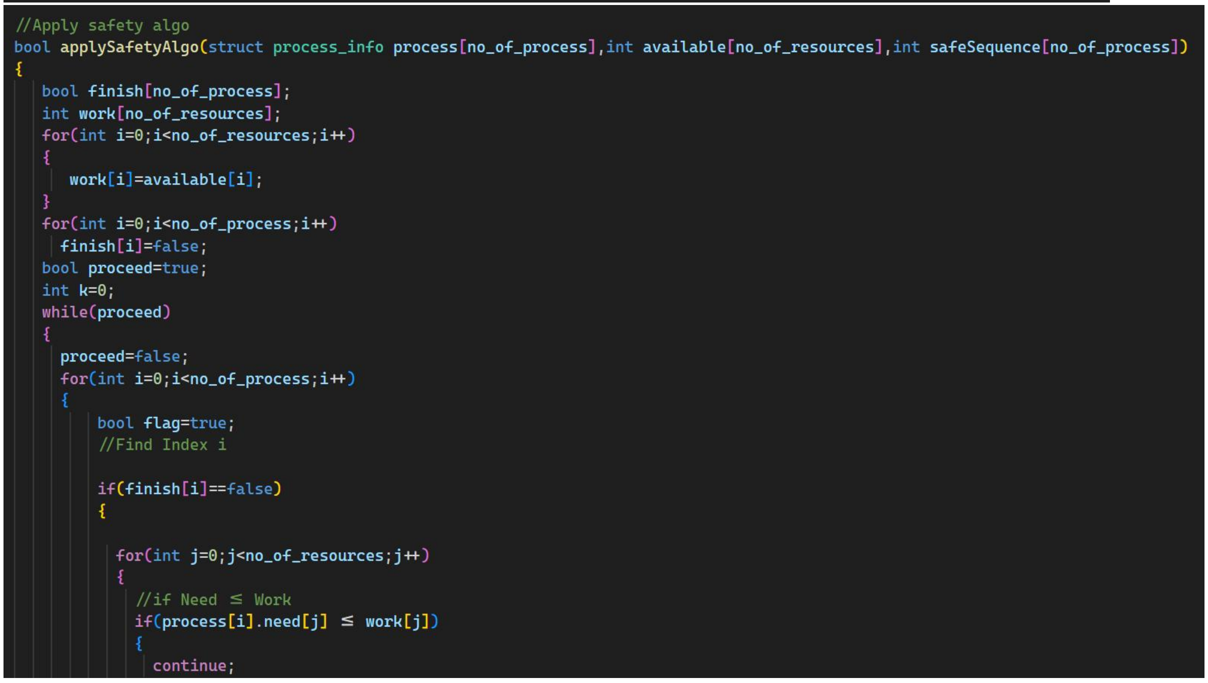
Code:



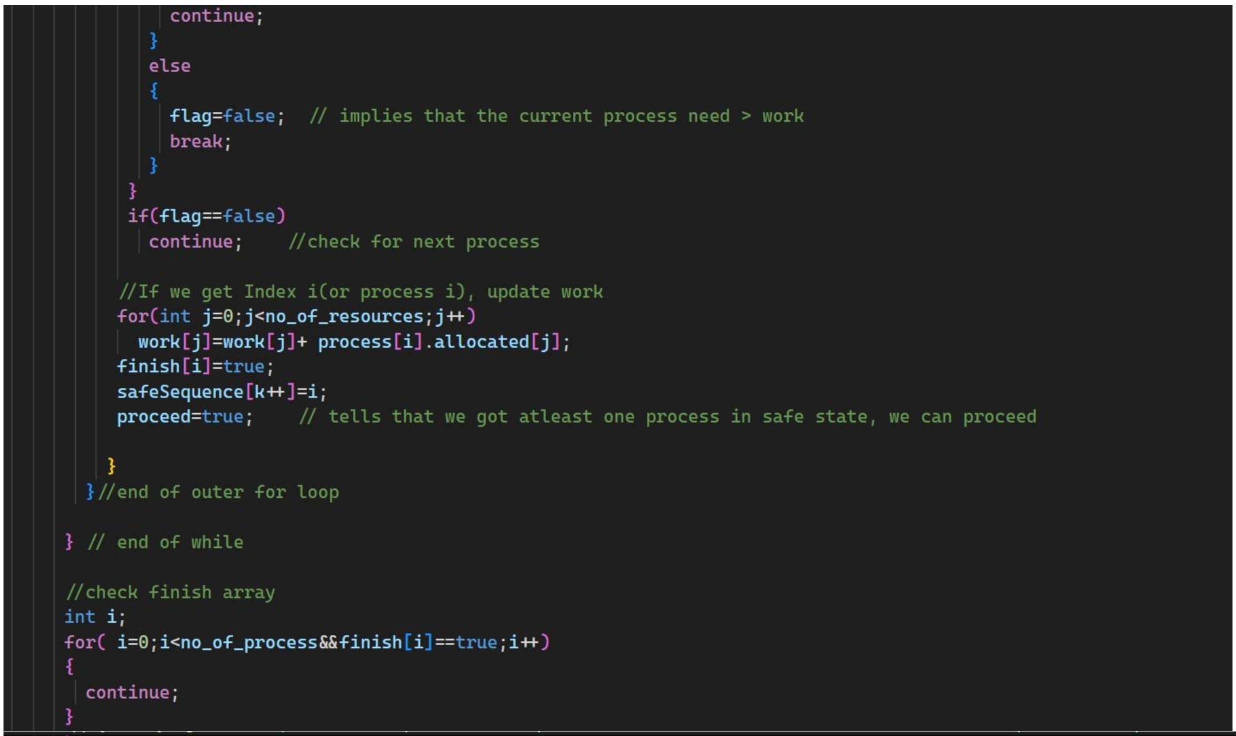
```c
#include <stdio.h>
#include <stdbool.h>

struct process_info
{
    int max[10];
    int allocated[10];
    int need[10];
};

int no_of_process,no_of_resources;

//Take the input
void input(struct process_info process[no_of_process],int available[no_of_resources])
{
    //Fill array of Structure
    for(int i=0;i<no_of_process;i++)
    {
      printf("Enter process[%d] info\n",i);
     printf("Enter Maximum Need: ");
      for(int j=0;j<no_of_resources;j++)
        scanf("%d",&process[i].max[j]);
     printf("Enter No. of Allocated Resources for this process: ");
      for(int j=0;j<no_of_resources;j++)
      {
        scanf("%d",&process[i].allocated[j]);
        //calculate need/future need
        process[i].need[j]= process[i].max[j] - process[i].allocated[j];
      }
    }
   printf("Enter Available Resources: ");
```



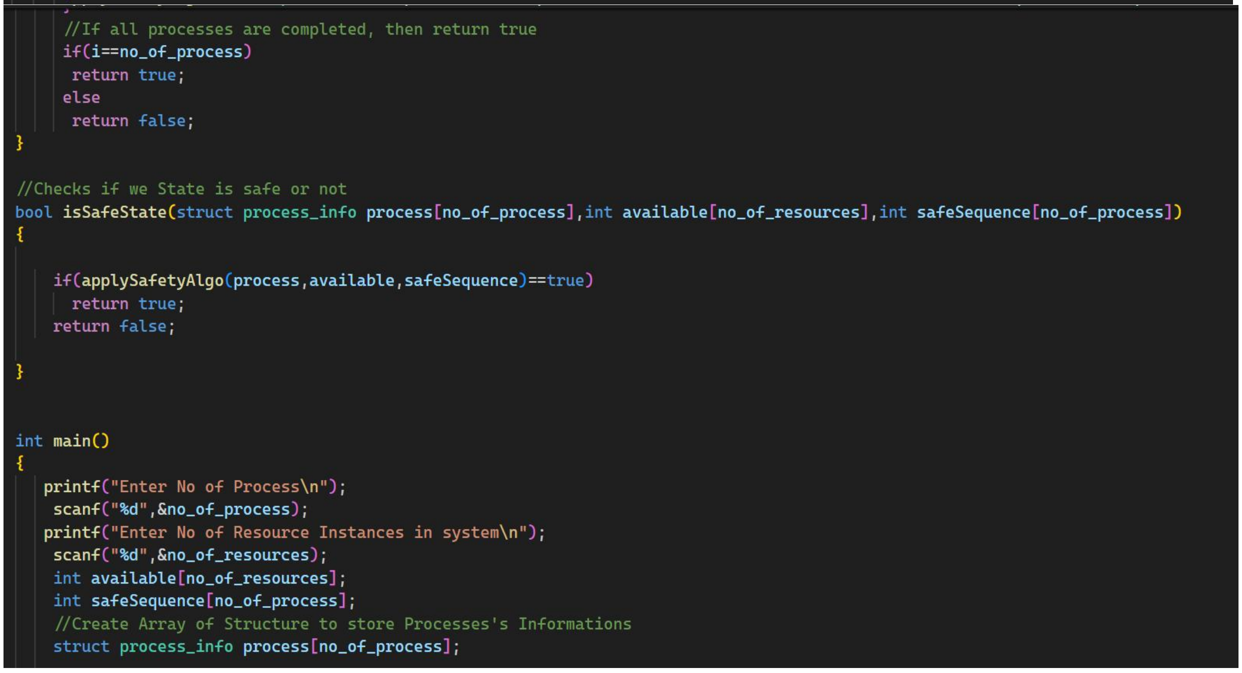
```c
    for(int i=0;i<no_of_resources;i++)
    {
        scanf("%d",&available[i]);
    }
}

//Print the Info in Tabular Form
void showTheInfo(struct process_info process[no_of_process])
{
    printf("\nPID\tMaximum\t\tAllocated\tNeed\n");
    for(int i=0;i<no_of_process;i++)
    {
        printf("P[%d]\t",i);
        for(int j=0;j<no_of_resources;j++)
            printf("%d ",process[i].max[j]);
        printf("\t\t");
        for(int j=0;j<no_of_resources;j++)
            printf("%d ",process[i].allocated[j]);
        printf("\t\t");
        for(int j=0;j<no_of_resources;j++)
            printf("%d ",process[i].need[j]);
        printf("\n");

    }
}
```



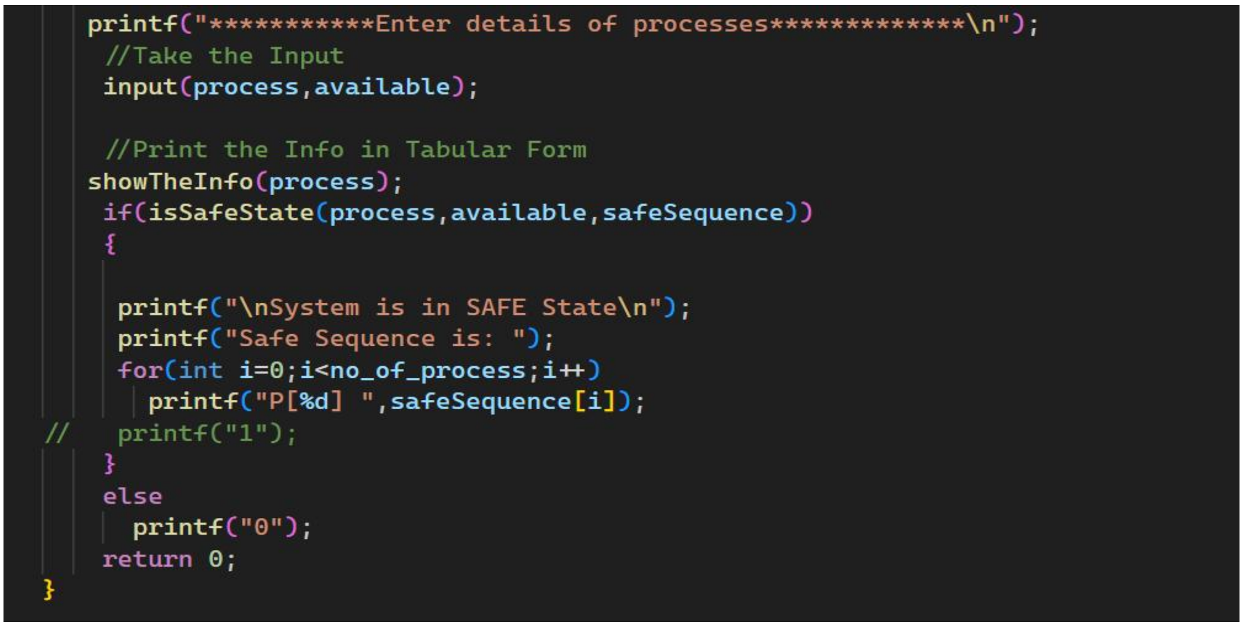
```c
//Apply safety algo
bool applySafetyAlgo(struct process_info process[no_of_process],int available[no_of_resources],int safeSequence[no_of_process])
{
    bool finish[no_of_process];
    int work[no_of_resources];
    for(int i=0;i<no_of_resources;i++)
    {
        work[i]=available[i];
    }
    for(int i=0;i<no_of_process;i++)
        finish[i]=false;
    bool proceed=true;
    int k=0;
    while(proceed)
    {
        proceed=false;
        for(int i=0;i<no_of_process;i++)
        {
            bool flag=true;
            //Find Index i

            if(finish[i]==false)
            {

                for(int j=0;j<no_of_resources;j++)
                {
                    //if Need <= Work
                    if(process[i].need[j] <= work[j])
                    {
                        continue;
```



```c
                  continue;
                }
                else
                {
                  flag=false;   // implies that the current process need > work
                  break;
                }
              }
              if(flag==false)
                continue;     //check for next process

             //If we get Index i(or process i), update work
             for(int j=0;j<no_of_resources;j++)
               work[j]=work[j]+ process[i].allocated[j];
             finish[i]=true;
             safeSequence[k++]=i;
             proceed=true;     // tells that we got atleast one process in safe state, we can proceed

            }
          }//end of outer for loop

        } // end of while

        //check finish array
        int i;
        for( i=0;i<no_of_process&&finish[i]==true;i++)
        {
          continue;
        }
```

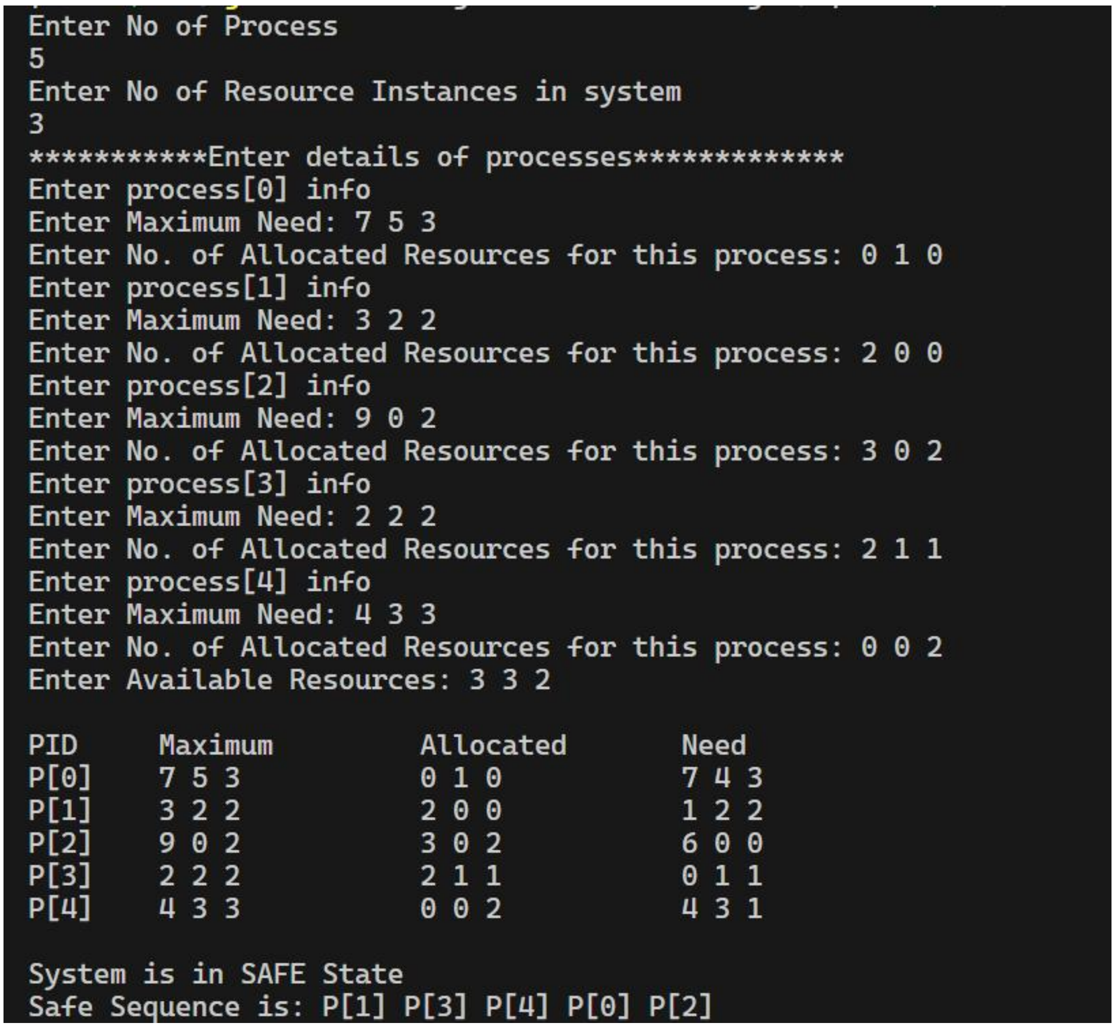


```c
    //If all processes are completed, then return true
    if(i==no_of_process)
     return true;
    else
     return false;
}

//Checks if we State is safe or not
bool isSafeState(struct process_info process[no_of_process],int available[no_of_resources],int safeSequence[no_of_process])
{

    if(applySafetyAlgo(process,available,safeSequence)==true)
      return true;
    return false;

}


int main()
{
   printf("Enter No of Process\n");
    scanf("%d",&no_of_process);
   printf("Enter No of Resource Instances in system\n");
    scanf("%d",&no_of_resources);
    int available[no_of_resources];
    int safeSequence[no_of_process];
    //Create Array of Structure to store Processes's Informations
    struct process_info process[no_of_process];
```



```c
    printf("***********Enter details of processes*************\n");
     //Take the Input
     input(process,available);

     //Print the Info in Tabular Form
    showTheInfo(process);
     if(isSafeState(process,available,safeSequence))
     {

      printf("\nSystem is in SAFE State\n");
      printf("Safe Sequence is: ");
      for(int i=0;i<no_of_process;i++)
        printf("P[%d] ",safeSequence[i]);
//    printf("1");
     }
     else
       printf("0");
     return 0;
}
```

Output:



```
Enter No of Process
5
Enter No of Resource Instances in system
3
***********Enter details of processes*************
Enter process[0] info
Enter Maximum Need: 7 5 3
Enter No. of Allocated Resources for this process: 0 1 0
Enter process[1] info
Enter Maximum Need: 3 2 2
Enter No. of Allocated Resources for this process: 2 0 0
Enter process[2] info
Enter Maximum Need: 9 0 2
Enter No. of Allocated Resources for this process: 3 0 2
Enter process[3] info
Enter Maximum Need: 2 2 2
Enter No. of Allocated Resources for this process: 2 1 1
Enter process[4] info
Enter Maximum Need: 4 3 3
Enter No. of Allocated Resources for this process: 0 0 2
Enter Available Resources: 3 3 2

PID     Maximum         Allocated       Need
P[0]    7 5 3           0 1 0           7 4 3
P[1]    3 2 2           2 0 0           1 2 2
P[2]    9 0 2           3 0 2           6 0 0
P[3]    2 2 2           2 1 1           0 1 1
P[4]    4 3 3           0 0 2           4 3 1

System is in SAFE State
Safe Sequence is: P[1] P[3] P[4] P[0] P[2]
```