# Name-Srikanth Reddy

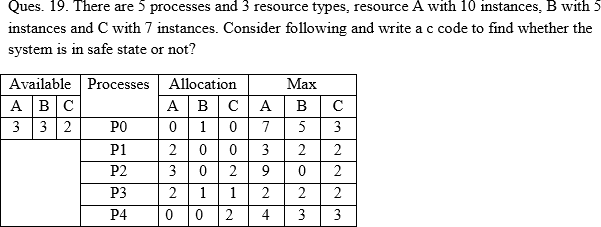
# Reg.No-11704097

# Roll.no-65

# Sec-K17GN

# Question No:19

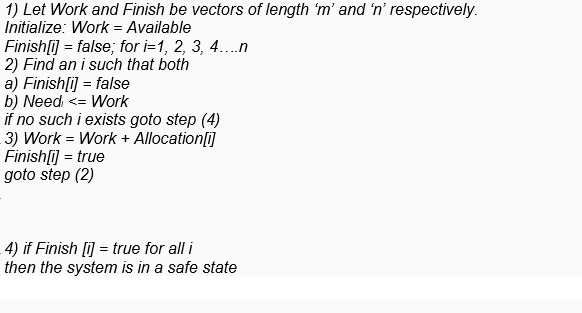
BANKER’S ALGORITHM :

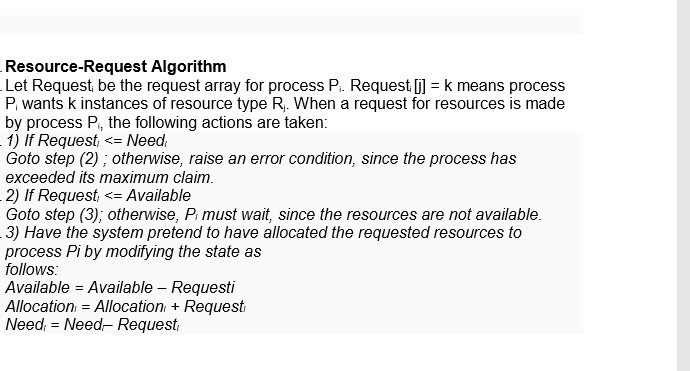


## DESCRIPTION AND UNDERSTANDING OF THE QUESTION :

The algorithm was developed in the design process for the operating system and originally described (in Dutch) in EWD108.[1] When a new process enters a system, it must declare the maximum number of instances of each resource type that it may ever claim; clearly, that number may not exceed the total number of resources in the system. Also, when a process gets all its requested resources it must return them in a finite amount of time.

## ALGORITHM :





Description (purpose of use):

The Banker algorithm, sometimes referred to as the detection algorithm, is a resource allocation and deadlock avoidance algorithm developed by Edsger Dijkstra that tests for safety by simulating the allocation of predetermined maximum possible amounts of all resources, and then

makes an "s-state" check to test for possible deadlock conditions for all other pending activities, before deciding whether allocation should be allowed to continue.

## CODE:

## #include<conio.h>

## #include<stdio.h>

## int main()

## {

## int n,r,i,j,k,cntt,cnt; //n number of process int r; // number of resources int i,j,k,cnt,cntt;

## 

## int avail[10],p[10];

## int need[10][10],alloc[10][10],max[10][10];

## printf("\nEnter number of process : "); scanf("%d",&n);

## printf("\n Enter resources available : "); scanf("%d",&r);

## printf("\nEnter insatnces for resources :\n");

## for(i=0;i<r;i++)

## {

## printf("R%d ",i+1);

## scanf("%d",&avail[i]);

## }

## printf("\n Enter allocation matrix \n"); for(i=0;i<n;i++)

## {

## printf("p%d",i+1); p[i]=0;

## for(j=0;j<r;j++)

## {

## scanf("%d",&alloc[i][j]);

## }

## 

## }

## printf("\n Enter MAX matrix \n"); for(i=0;i<n;i++)

## {

## printf("p%d",i+1); for(j=0;j<r;j++)

## {

## scanf("%d",&max[i][j]);

## }

## }

## for(i=0;i<n;i++)

## {

## printf("\np%d\t",i+1) ; for(j=0;j<r;j++)

## {

## need[i][j]=max[i][j]-alloc[i][j];

## printf("\t%d",need[i][j]);

## }

## }

## k=0;

## cntt=0; printf("\n\n");

## 

## while(k<15)

## {

## for(i=0;i<n;i++)

## { cnt=0; for(j=0;j<r;j++)

## {

## if(p[i]==1) break; if(need[i][j]<=avail[j])

## {

## cnt++;

## }

## if(cnt==r)

## {

## for(j=0;j<r;j++)

## {

## avail[j]+=alloc[i][j];

## }

## printf("p%d\t",i+1); p[i]=1; cntt++;

## }

## }

## } k++;

## }

## if(cntt<n-1)

## {

## printf("\n deadlock ");

## 

## }

## getch();

## }

## OUTPUT :

