**Name:** Chillara V L N S Pavana Vamsi

**Reg.no:** 21BCE5095

**Date:** 09/2/2023

**Faculty:** M Sivagami

## 

## L21\_L22\_Lab7 Exercises

### Implement the banker's algorithm for n processes with m resources. Show the execution of your C program using suitable data set

### with deadlock and

### without dead lock.

#include*<stdio.h>*

#include*<stdbool.h>*

#define row 5

#define col 3

bool final[row];

void initailFinal()

{

for(int i=0;i<row;i++)

final[i]=false;

}

void pout(int *a*[row][col])

{

for(int i=0;i<row;i++)

{

for(int j=0;j<col;j++)

printf(*"%d "*,*a*[i][j]);

printf(*"\n"*);

}

printf(*"\n"*);

}

void pr(int *a*[col])

{

for(int i=0;i<col;i++)

{

printf(*"%d "*,*a*[i]);

}

}

int main()

{

int allocation[row][col]={{10,1,0},{2,0,0},{3,0,22},{2,1,1},{10,0,2}};

printf(*"Allocation Matrix:\n"*);

pout(allocation);

int max[row][col]={{17,5,3},{3,2,2},{9,0,22},{2,2,2},{16,3,3}};

printf(*"Maximum Resource Matrix:\n"*);

pout(max);

int request[row][col];

int available[col]={3,3,2};

int cavailable[col];

int x=row;

for(int i=0;i<row;i++)

for(int j=0;j<col;j++)

request[i][j]=max[i][j]-allocation[i][j];

printf(*"Request matrix:\n"*);

pout(request);

initailFinal();

for(int i=0;i<col;i++)

{

cavailable[i]=available[i];

}

int j=0,count=0;

for(int k=0;k<row;k++)

*//while(x!=0)*

{

for(int i=0;i<row;i++)

{

if(final[i]==false)

{

        count=0;

        for(int j=0;j<col;j++)

        {

                if(cavailable[j]>=request[i][j])

                {

                        count++;

                }

        }

        if(count==col)

        {

*//printf("P[%d]->",i+1);*

                for(int j=0;j<col;j++)

                {

                cavailable[j]+=allocation[i][j];

                }

                final[i]=true;

                x--;

        }

}

}

}

for(int i=0;i<row;i++)

{

if(final[i]==0)

x=1;

}

if(x!=1)

{

printf(*"No Dead lock\nTotal resources left:"*);

*//pr(cavailable);*

}

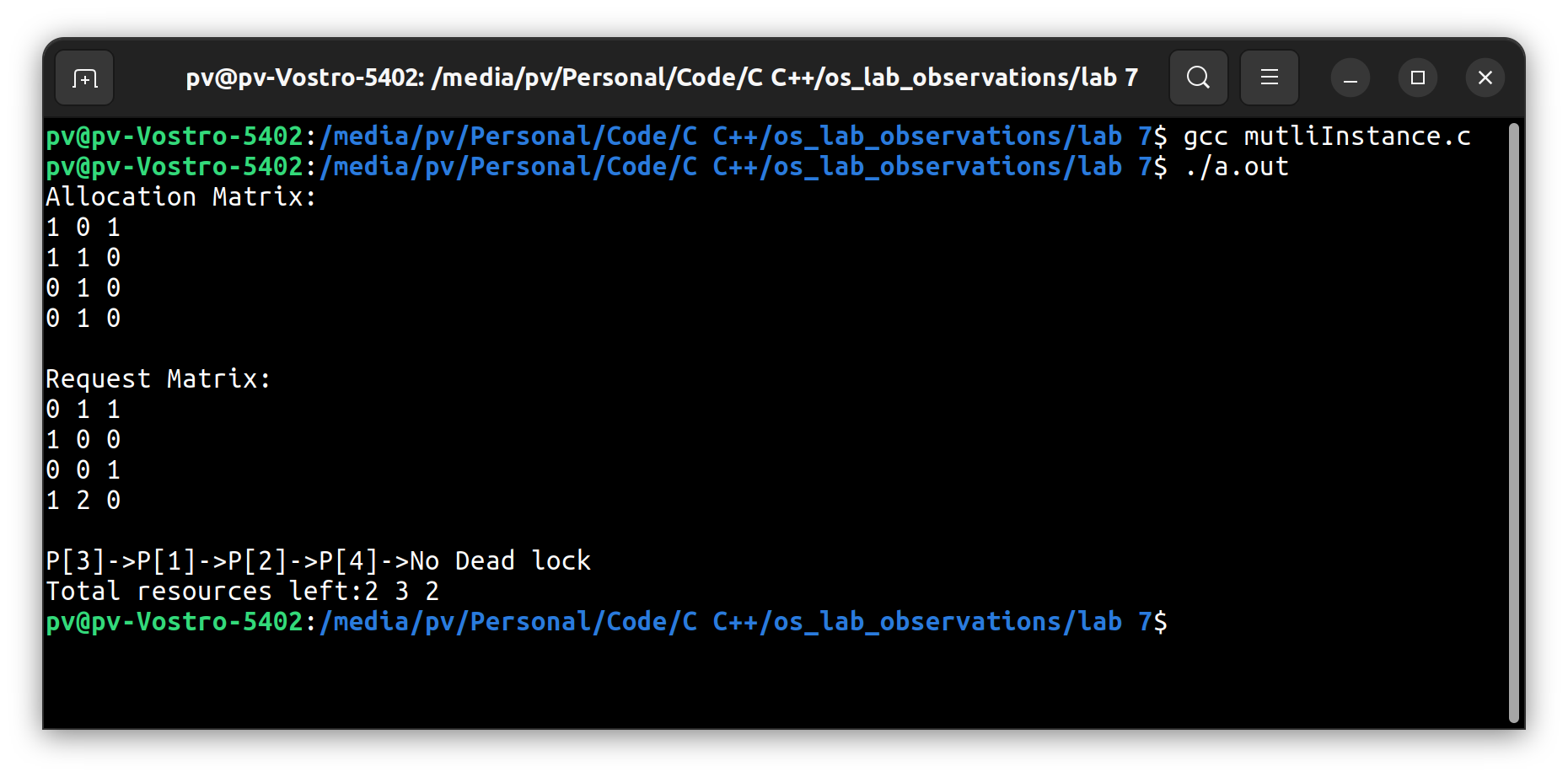
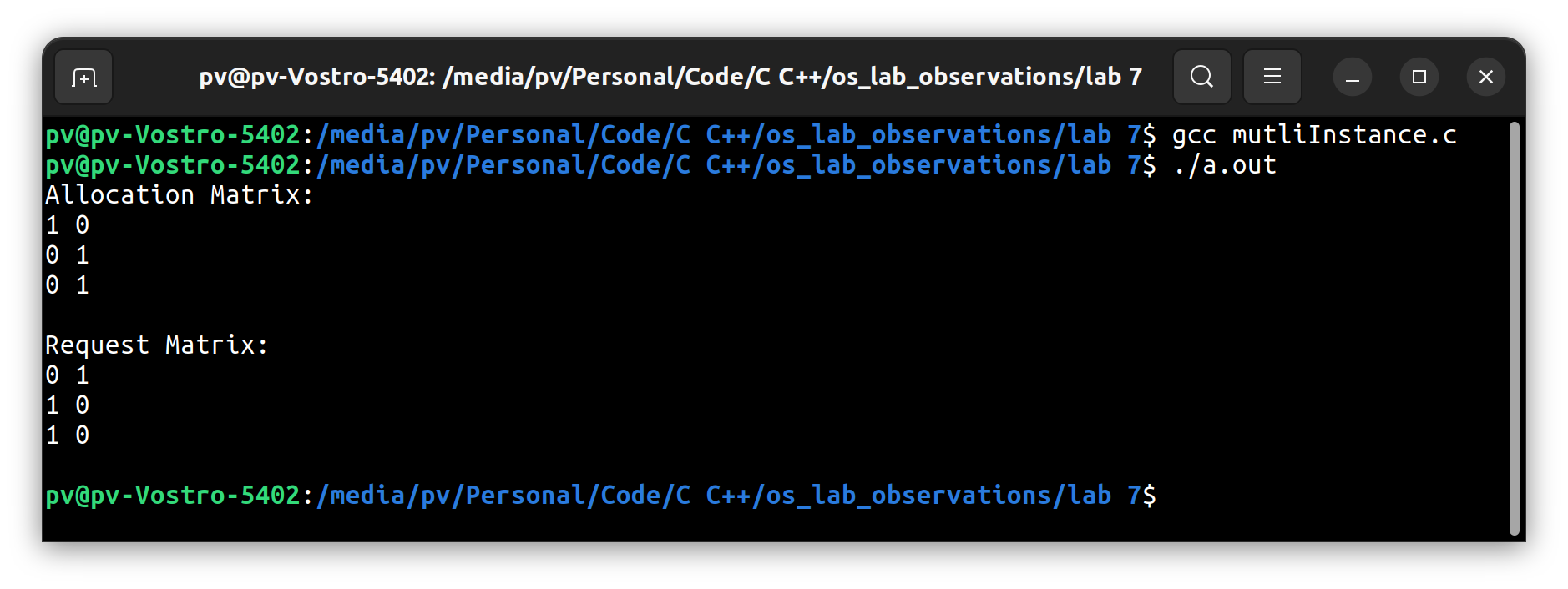
else

printf(*"Dead lock occurs"*);

*//pout(request);*

return 0;

}

1. Develop the C program to check whether there is a deadlock or not from Multiple Instance Resource Allocation Graph.

#include*<stdio.h>*

#include*<stdbool.h>*

#define row 3

#define col 2

bool final[row];

void initailFinal()

{

for(int i=0;i<row;i++)

final[i]=false;

}

void pout(int *a*[row][col])

{

for(int i=0;i<row;i++)

{

for(int j=0;j<col;j++)

printf(*"%d "*,*a*[i][j]);

printf(*"\n"*);

}

printf(*"\n"*);

}

void pr(int *a*[col])

{

for(int i=0;i<col;i++)

{

printf(*"%d "*,*a*[i]);

}

printf(*"\n"*);

}

int main()

{

int allocation[row][col]={{1,0},{0,1},{0,1}};

printf(*"Allocation Matrix:\n"*);

pout(allocation);

int request[row][col]={{0,1},{1,0},{1,0}};

printf(*"Request Matrix:\n"*);

pout(request);

int available[col]={0,0};

int cavailable[col];

int x=row;

initailFinal();

for(int i=0;i<col;i++)

{

cavailable[i]=available[i];

}

int count=0;

for(int k=0;k<row;k++)

*//while(x!=0)*

{

for(int i=0;i<row;i++)

{

if(final[i]==false)

{

        count=0;

        for(int j=0;j<col;j++)

        {

                if(cavailable[j]>=request[i][j])

                {

                        count++;

                }

        }

        if(count==col)

        {

                printf(*"P[%d]->"*,i+1);

                for(int j=0;j<col;j++)

                {

                cavailable[j]+=allocation[i][j];

                }

                final[i]=true;

                x--;

        }

}

}

}

for(int i=0;i<row;i++)

{

if(final[i]==0)

x=1;

}

if(x!=1)

{

printf(*"No Dead lock\nTotal resources left:"*);

pr(cavailable);

}

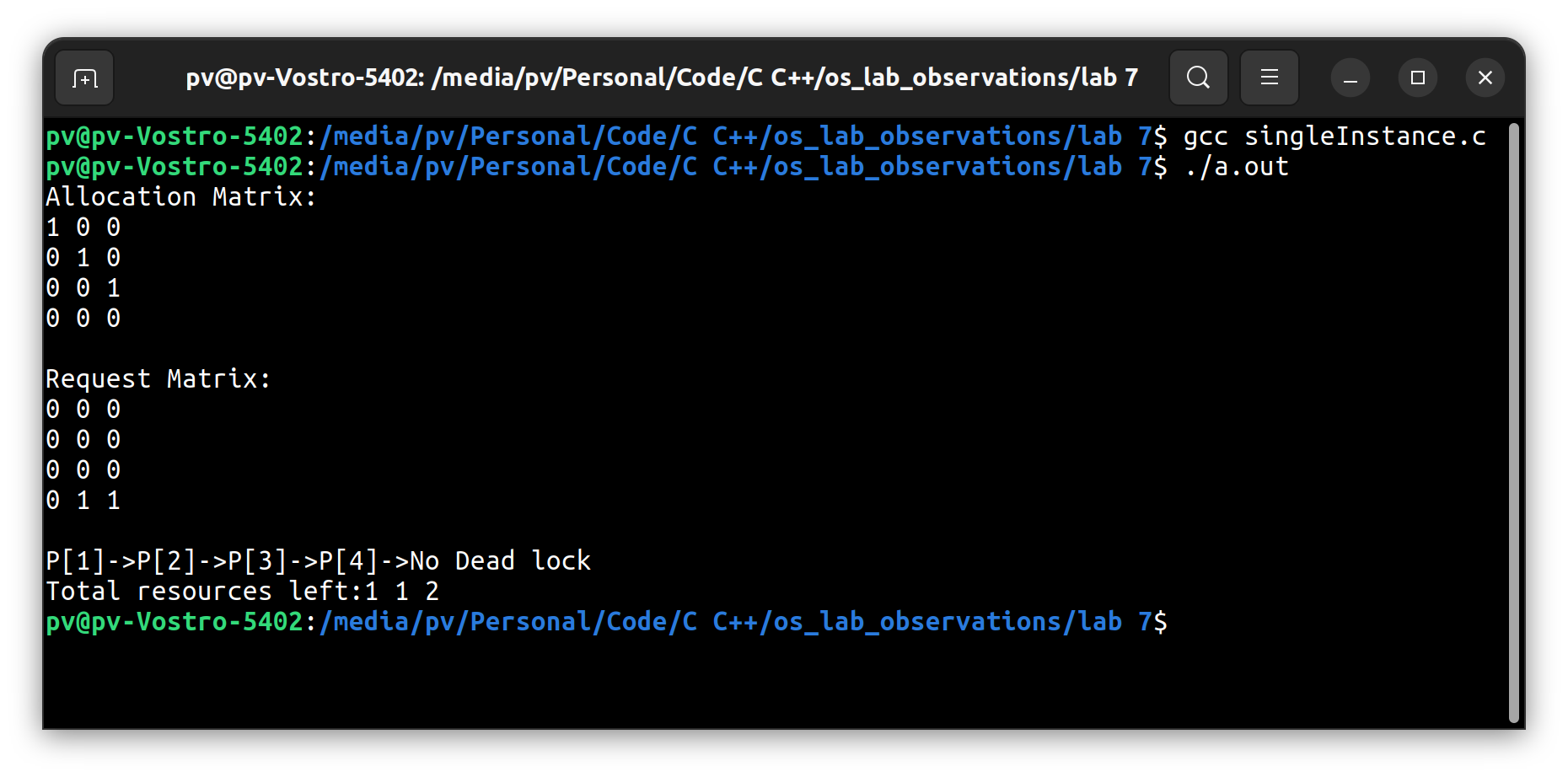
else

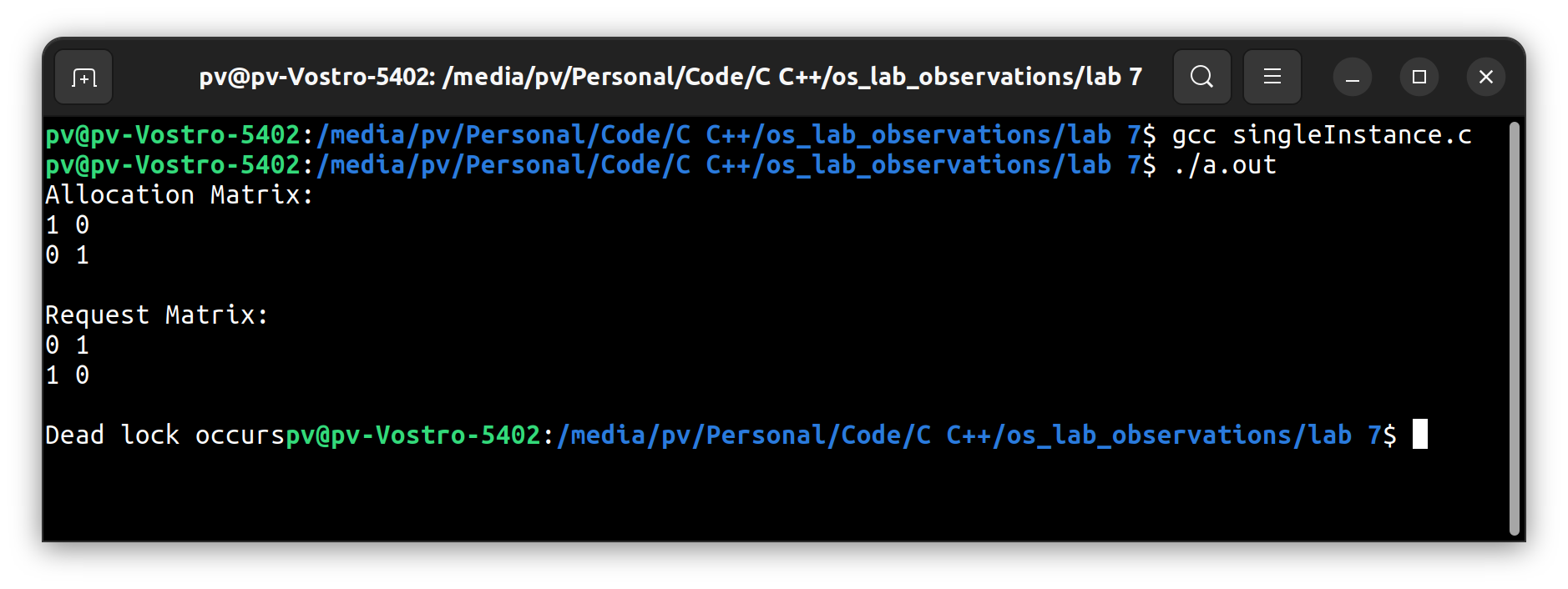
printf(*"Dead lock occurs"*);

*//pout(request);*

return 0;

}

1.  Develop the C program to check whether there is a deadlock or not from Single Instance Resource Allocation Graph.



#include*<stdio.h>*

#include*<stdbool.h>*

#define row 2

#define col 2

bool final[row];

void initailFinal()

{

        for(int i=0;i<row;i++)

        final[i]=false;

}

void pout(int *a*[row][col])

{

        for(int i=0;i<row;i++)

        {

                for(int j=0;j<col;j++)

                printf(*"%d "*,*a*[i][j]);

                printf(*"\n"*);

        }

        printf(*"\n"*);

}

void pr(int *a*[col])

{

        for(int i=0;i<col;i++)

        {

                printf(*"%d "*,*a*[i]);

        }

        printf(*"\n"*);

}

int main()

{

int allocation[row][col]={{1,0},{0,1}};

printf(*"Allocation Matrix:\n"*);

pout(allocation);

int request[row][col]={{0,1},{1,0}};

printf(*"Request Matrix:\n"*);

pout(request);

int available[col]={0,0};

int cavailable[col];

int x=row;

initailFinal();

for(int i=0;i<col;i++)

{

cavailable[i]=available[i];

}

int count=0;

for(int k=0;k<row;k++)

*//while(x!=0)*

{

for(int i=0;i<row;i++)

{

if(final[i]==false)

{

        count=0;

        for(int j=0;j<col;j++)

        {

                if(cavailable[j]>=request[i][j])

                {

                        count++;

                }

        }

        if(count==col)

        {

                printf(*"P[%d]->"*,i+1);

                for(int j=0;j<col;j++)

                {

                cavailable[j]+=allocation[i][j];

                }

                final[i]=true;

                x--;

        }

}

}

}

for(int i=0;i<row;i++)

{

if(final[i]==0)

x=1;

}

if(x!=1)

{

printf(*"No Dead lock\nTotal resources left:"*);

pr(cavailable);

}

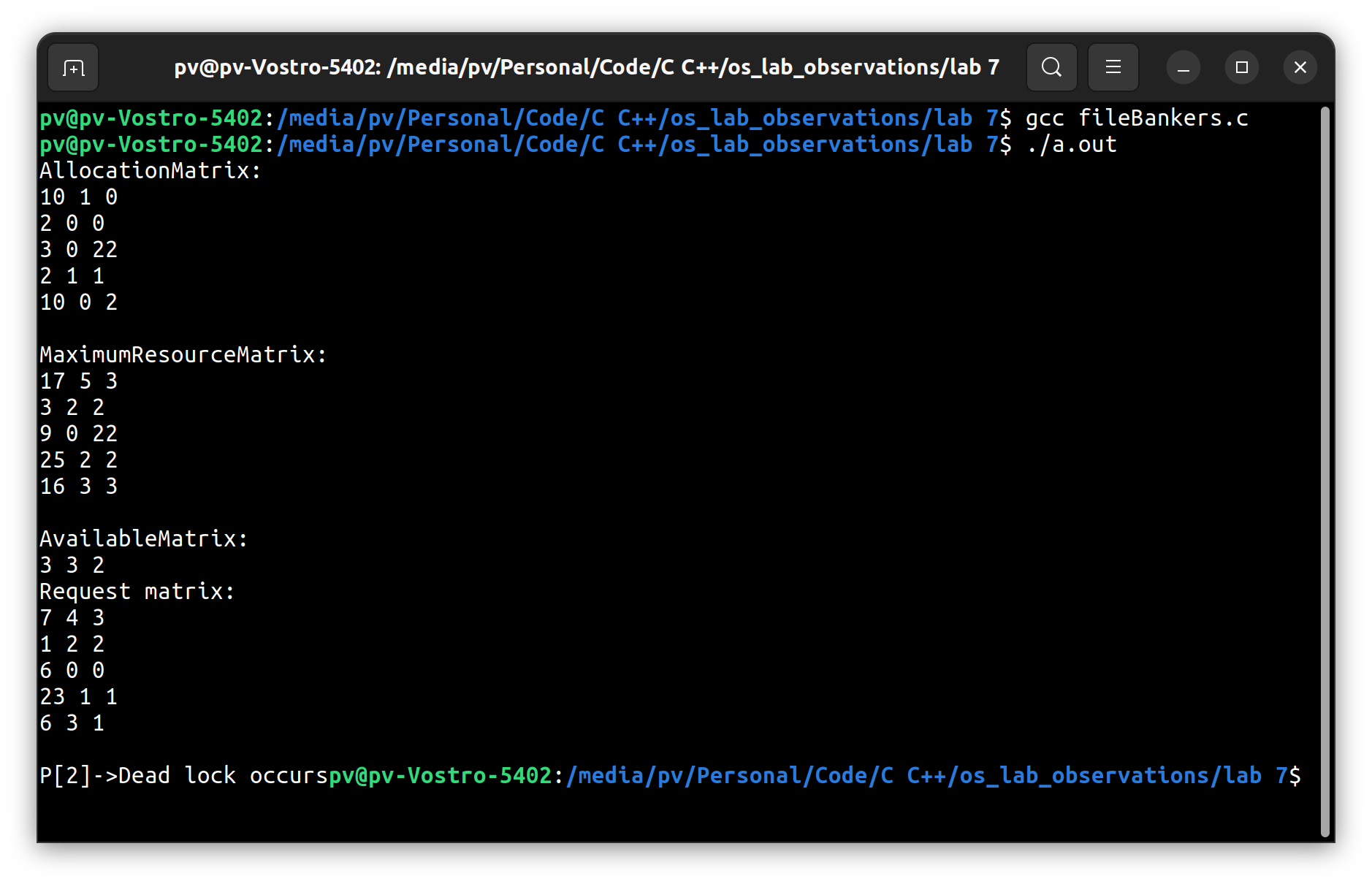
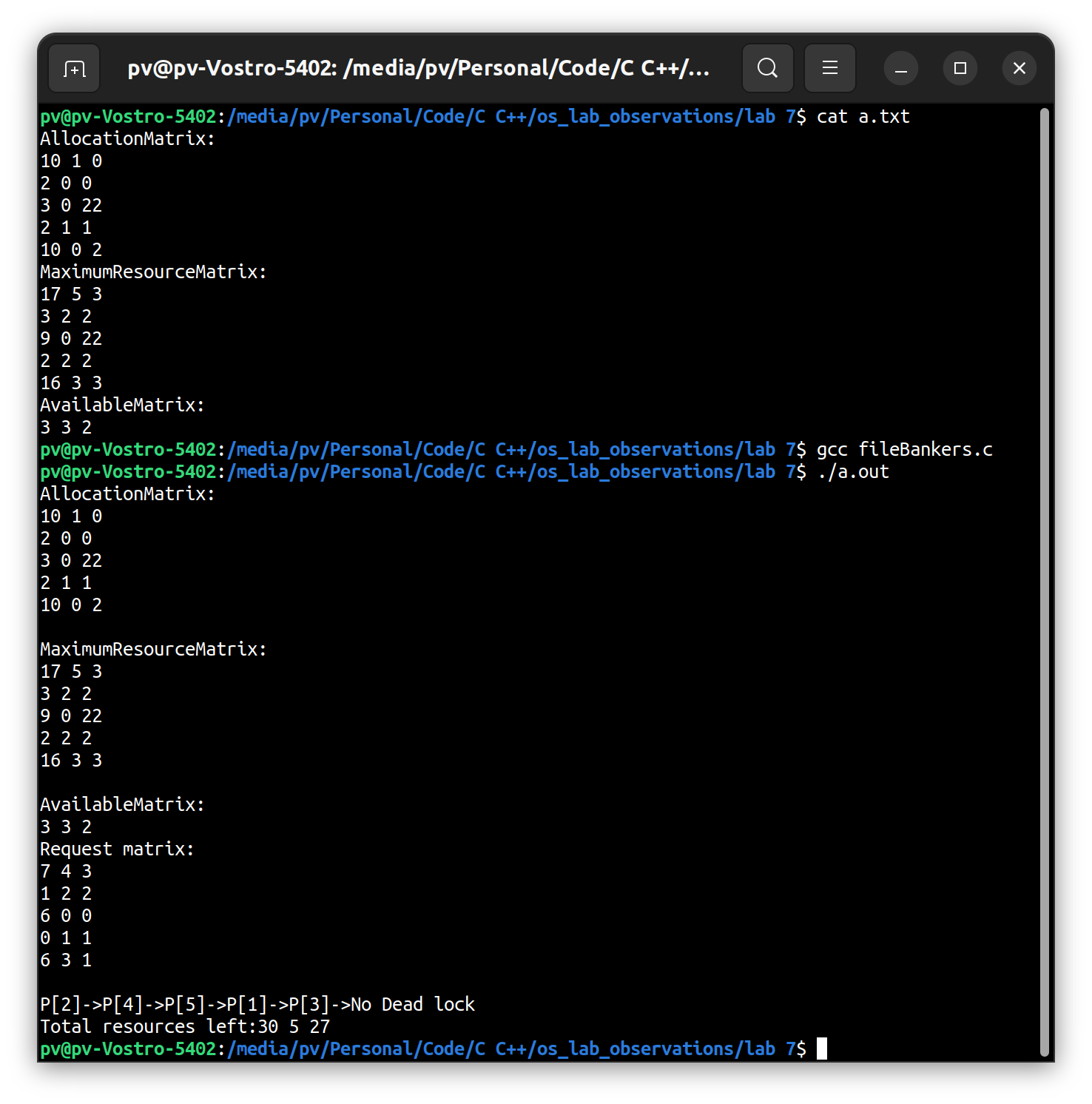
else

printf(*"Dead lock occurs"*);

*//pout(request);*

return 0;

}

1. Modify the question to read the data from the file "a.txt" and implement the banker's algorithm for the same.

#include*<stdio.h>*

#include*<stdbool.h>*

#define row 5

#define col 3

bool final[row];

void initailFinal()

{

for(int i=0;i<row;i++)

final[i]=false;

}

void pout(int *a*[row][col])

{

for(int i=0;i<row;i++)

{

for(int j=0;j<col;j++)

printf(*"%d "*,*a*[i][j]);

printf(*"\n"*);

}

printf(*"\n"*);

}

void pr(int *a*[col])

{

for(int i=0;i<col;i++)

{

printf(*"%d "*,*a*[i]);

}

printf(*"\n"*);

}

int main()

{

char  name[30];

FILE \*fp=fopen(*"a.txt"*,*"r"*);

int allocation[row][col];

fscanf(fp,*"%s"*,name);

printf(*"%s\n"*,name);

for(int i=0;i<row;i++)

for(int j=0;j<col;j++)

fscanf(fp,*"%d"*,&allocation[i][j]);

pout(allocation);

fscanf(fp,*"%s"*,name);

int max[row][col];

for(int i=0;i<row;i++)

for(int j=0;j<col;j++)

fscanf(fp,*"%d"*,&max[i][j]);

printf(*"%s\n"*,name);

pout(max);

int request[row][col];

fscanf(fp,*"%s"*,name);

printf(*"%s\n"*,name);

int available[col];

for(int i=0;i<col;i++)

fscanf(fp,*"%d"*,&available[i]);

pr(available);

fclose(fp);

int cavailable[col];

int x=row;

for(int i=0;i<row;i++)

for(int j=0;j<col;j++)

request[i][j]=max[i][j]-allocation[i][j];

printf(*"Request matrix:\n"*);

pout(request);

initailFinal();

for(int i=0;i<col;i++)

{

cavailable[i]=available[i];

}

int j=0,count=0;

for(int k=0;k<row;k++)

*//while(x!=0)*

{

for(int i=0;i<row;i++)

{

if(final[i]==false)

{

        count=0;

        for(int j=0;j<col;j++)

        {

                if(cavailable[j]>=request[i][j])

                {

                        count++;

                }

        }

        if(count==col)

        {

                printf(*"P[%d]->"*,i+1);

                for(int j=0;j<col;j++)

                {

                cavailable[j]+=allocation[i][j];

                }

                final[i]=true;

                x--;

        }

}

}

}

for(int i=0;i<row;i++)

{

if(final[i]==0)

x=1;

}

if(x!=1)

{

printf(*"No Dead lock\nTotal resources left:"*);

pr(cavailable);

}

else

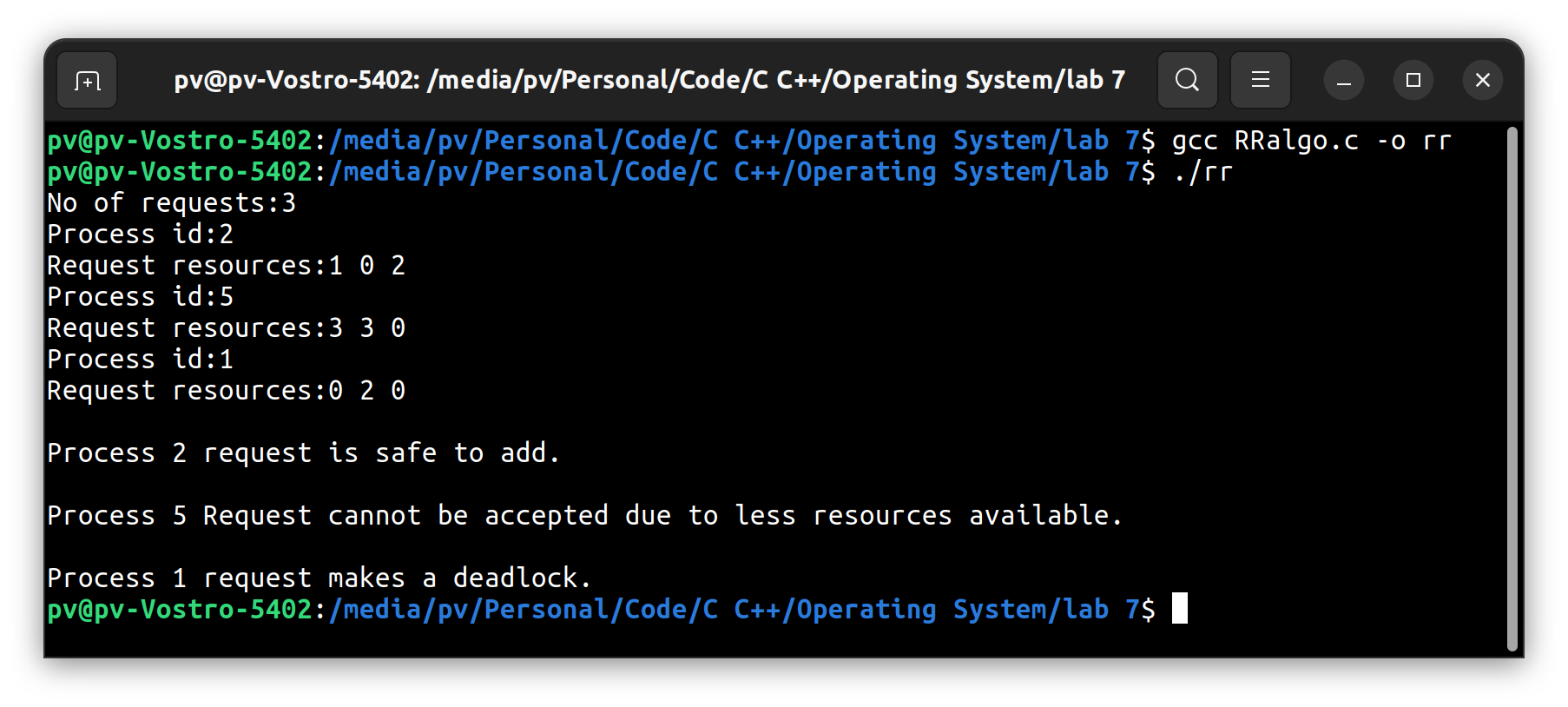
printf(*"Dead lock occurs"*);

*//pout(request);*

return 0;

}

1. Resource Request Algorithm:



#include*<stdio.h>*

#include*<stdbool.h>*

#define row 5

#define col 3

void initailFinal(bool *final*[])

{

for(int i=0;i<row;i++)

*final*[i]=false;

}

void print\_2d(int *a*[row][col])

{

        for(int i=0;i<row;i++)

        {

                for(int j=0;j<col;j++)

                        printf(*"%d "*,*a*[i][j]);

                printf(*"\n"*);

        }

        printf(*"\n"*);

}

void print\_1d(int *a*[col])

{

for(int i=0;i<col;i++)

        printf(*"%d "*,*a*[i]);

}

int compare(int *a*[col],int *b*[col])

{

        int x=0;

                for(int i=0;i<col;i++)

                {

                if(*a*[i]<=*b*[i])

x++;

                }

                if(x==col)

                return 1;

                else

                return 0;

}

int bankers(int *allocation*[row][col],int *request*[row][col],int *available*[col])

{

        int cavailable[col];

        int x=row;

        bool final[row];

                initailFinal(final);

                for(int i=0;i<col;i++)

                    cavailable[i]=*available*[i];

        int j=0,count=0;

                for(int k=0;k<row;k++)

                {

                    for(int i=0;i<row;i++)

                    {

                        if(final[i]==false)

                        {

                            count=0;

                            for(int j=0;j<col;j++)

                            {

                                if(cavailable[j]>=*request*[i][j])

                                    count++;

                            }

                            if(count==col)

                            {

                                for(int j=0;j<col;j++)

                                {

                                cavailable[j]+=*allocation*[i][j];

                                }

                                final[i]=true;

                                x--;

                            }

                        }

                    }

                }

                for(int i=0;i<row;i++)

                {

                if(final[i]==0)

                x=1;

                }

                if(x!=1)

                {

                return 0;

                }

                else

                return 1;

}

void process(int *x*,int *resources\_request*[][col],int *allocation*[row][col],int *request*[row][col],int *available*[col],int *k*)

{

        int t;

        int new\_allocation[row][col],new\_request[row][col],new\_available[col];

                for(int i=0;i<row;i++)

                {

                        for(int j=0;j<col;j++)

                        {

                        new\_allocation[i][j]=*allocation*[i][j];

                        new\_request[i][j]=*request*[i][j];

                        new\_available[j]=*available*[j];

                        }

                }

        if(compare(*resources\_request*[*k*],new\_available))

                {

                for(int i=0;i<col;i++)

                {

                t=*resources\_request*[*k*][i];

                new\_allocation[*x*][i]=new\_allocation[*x*][i]+t;

                new\_available[i]=new\_available[i]-t;

                new\_request[*x*][i]=new\_request[*x*][i]-t;

                }

        if(bankers(new\_allocation,new\_request,new\_available)==0)

        {

                printf(*"\nProcess %d request is safe to add.\n"*,*x*+1);

                for(int i=0;i<col;i++)

                {

                t=*resources\_request*[*k*][i];

*allocation*[*x*][i]=*allocation*[*x*][i]+t;

*available*[i]=*available*[i]-t;

*request*[*x*][i]=*request*[*x*][i]-t;

                }

        }

                else

                printf(*"\nProcess %d request makes a deadlock.\n"*,*x*+1);

                }

                else

                printf(*"\nProcess %d Request cannot be accepted due to less resources available.\n"*,*x*+1);

}

void resourceRequest(int *allocation*[row][col],int *request*[row][col],int *available*[col])

{

        int n,x;

                printf(*"No of requests:"*);

                scanf(*"%d"*,&n);

        int process\_request[n];

        int resources\_request[n][col];

                for(int i=0;i<n;i++)

                {

                printf(*"Process id:"*);

                scanf(*"%d"*,&process\_request[i]);

                printf(*"Request resources:"*);

                for(int j=0;j<col;j++)

                scanf(*"%d"*,&resources\_request[i][j]);

                }

                for(int i=0;i<n;i++)

                {

                x=process\_request[i]-1;

        process(x,resources\_request,*allocation*,*request*,*available*,i);

                }

}

int main()

{

        int allocation[row][col]={{10,1,0},{2,0,0},{3,0,22},{2,1,1},{10,0,2}};

*//printf("Allocation Matrix:\n");*

*//print\_2d(allocation);*

        int max[row][col]={{17,5,3},{3,2,2},{9,0,22},{2,2,2},{16,3,3}};

*//printf("Maximum Resource Matrix:\n");*

*//print\_2d(max);*

        int request[row][col];

        int available[col]={3,3,2};

                for(int i=0;i<row;i++)

                for(int j=0;j<col;j++)

                request[i][j]=max[i][j]-allocation[i][j];

        resourceRequest(allocation,request,available);

return 0;

}