**Q5a)** **Implement menu-driven program for page replacement using MFU and LFU.**

**Consider the total number of references made by CPU are 16. Take page reference**

**string and number of page frames as run time input. Compute and display number of**

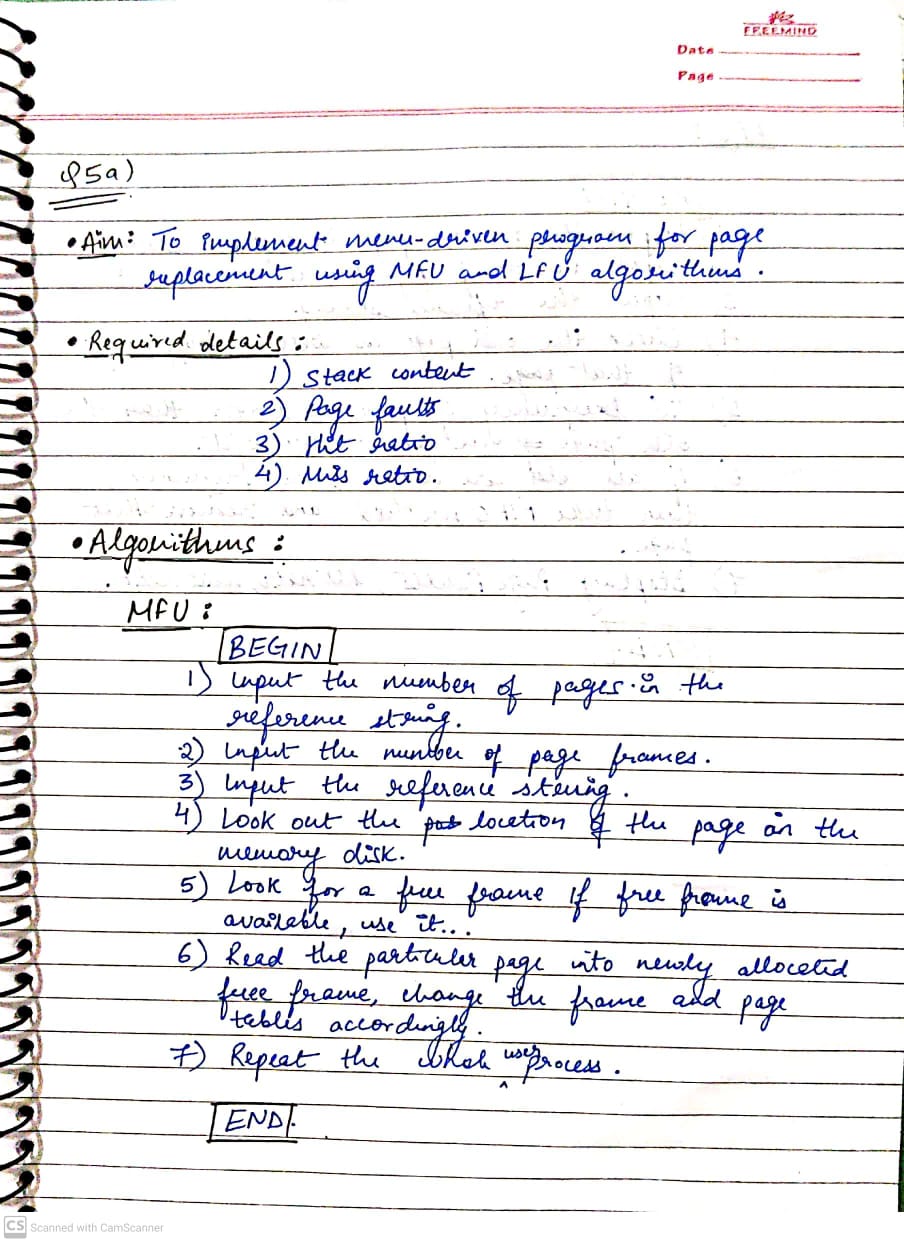
**page faults, hit ratio and miss ratio. Use stack data structure to record most recent**

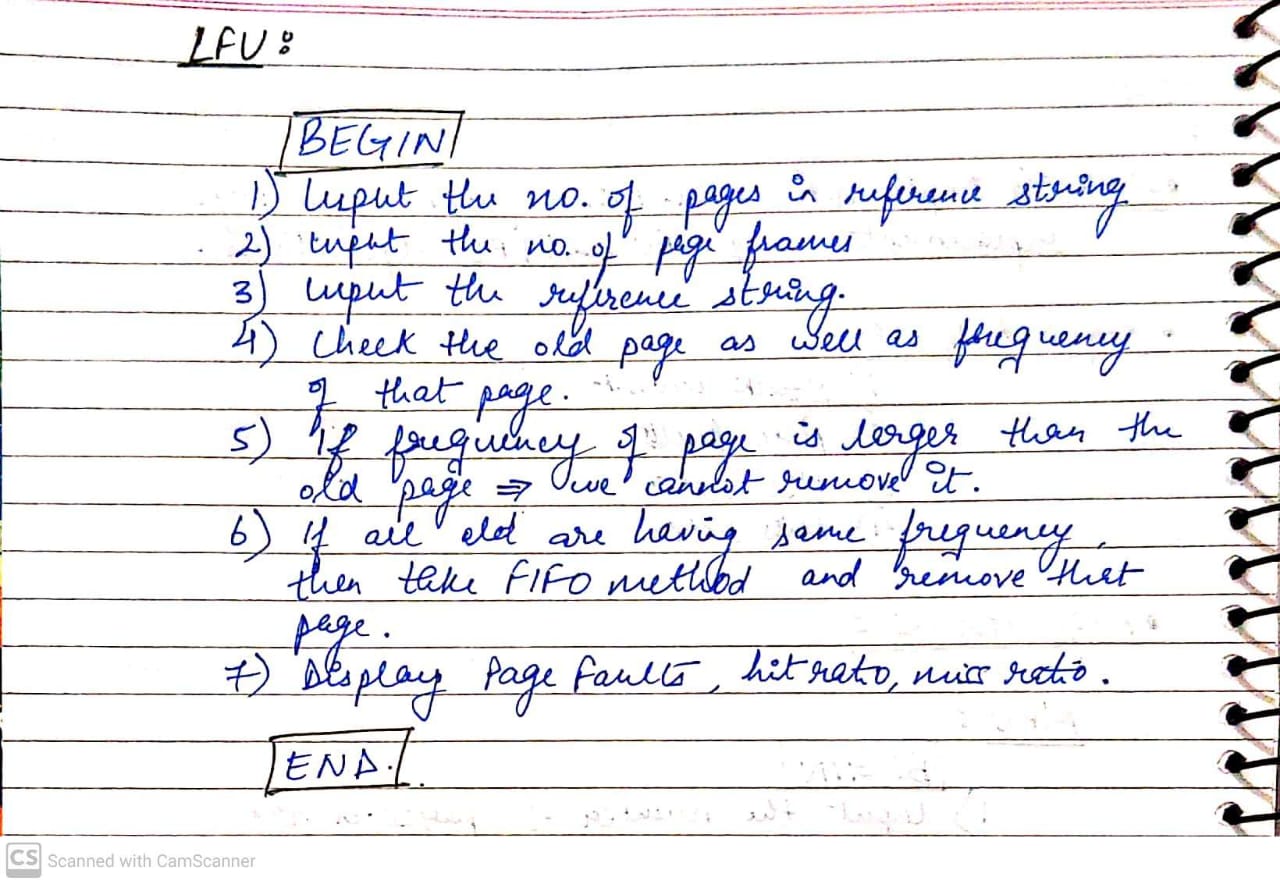
**page references. Assume initially all 3 page frames are empty. Display stack content**

**and number of page faults.**

**A5a)**

**Handwritten file:**

****

****

**Code:**

#include<stdio.h>

#include<stdlib.h>

int lfu();

int mfu()

{

int hit=0,miss=0,i,j,noPages,noFrames,min;

int frames[10],pages[20];

int flag=0,flag1=0,flag2=0;

int flagFound=0;

int count=0;

int frameAge[50],frameFREQ[50];

printf("enter number of frames\n");

scanf("%d",&noFrames);

printf("enter number of pages\n");

scanf("%d",&noPages);

printf("enter the page string ");

for(i=0;i<noPages;i++)

{

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

}

for(i=0;i<noFrames;i++)

{

frames[i]=-1;

frameAge[i]=-1;

}

for(j=0;j<noFrames;j++)

frameFREQ[j]=0;

for(j=0;j<noPages;j++)

{

int index;

printf(" page:%d ",pages[j] );

flagFound=0,flag=0,flag2=0;

for(i=0;i<noFrames;i++)

{

if(frames[i]==pages[j])

{

flagFound=1;

flag=1;

index=i;

printf("hit ");

hit++;

break;

}

}

if(flagFound==0) //if frame not found and empty frame avalible

{

for(i=0;i<noFrames;i++)

{

if(frames[i]==-1)

{

frames[i]=pages[j];

flag=1;

count++;

frameAge[i]=count;

miss++;

frameFREQ[i]=1;

printf("miss ");

break;

}

}

if(flag==0)

{

int bestmfu=0;

for(i=0;i<noFrames;i++)

{

if(frameFREQ[i]>frameFREQ[bestmfu])

bestmfu=i;

}

frames[bestmfu]=pages[j];

miss++;

printf("miss ");

frameFREQ[bestmfu]=1;

}

} //FLAG FOUND ends

else

{

frameFREQ[index]++;

}

for(i=0;i<noFrames;i++)

{

printf(" %d ",frames[i]);

}

printf("\n");

}

//printf("number of hits %d\n",hit);

//printf("number of miss %d\n",miss);

float faults;

faults=noPages-hit;

printf("\n\nPage Faults:\t%d\n",faults);

printf("Page Hit:\t%d\n",hit);

//printf("Page Miss:\t%d\n",miss);

float hitratio;

float hitt=hit;

float pagess=noPages;

hitratio=hitt/pagess;

printf("Hit Ratio:\t%.2f\n",hitratio);

float missratio;

missratio=1-hitratio;

printf("Miss Ratio:\t%.2f\n\n\n",missratio);

}

int lfu()

{

int total\_frames, total\_pages, hit = 0;

int pages[25], frame[10], arr[25], time[25];

int m, n, page, flag, k, minimum\_time, temp;

printf("Enter Total Number of Pages: ");

scanf("%d", &total\_pages);

printf("Enter Total Number of Frames: ");

scanf("%d", &total\_frames);

for(m = 0; m < total\_frames; m++)

{

frame[m] = -1;

}

for(m = 0; m < 25; m++)

{

arr[m] = 0;

}

printf("Enter Values of Reference String\n");

for(m = 0; m < total\_pages; m++)

{

printf("Enter Value No.[%d]:\t", m + 1);

scanf("%d", &pages[m]);

}

printf("\n");

for(m = 0; m < total\_pages; m++)

{

arr[pages[m]]++;

time[pages[m]] = m;

flag = 1;

k = frame[0];

for(n = 0; n < total\_frames; n++)

{

if(frame[n] == -1 || frame[n] == pages[m])

{

if(frame[n] != -1)

{

hit++;

}

flag = 0;

frame[n] = pages[m];

break;

}

if(arr[k] > arr[frame[n]])

{

k = frame[n];

}

}

if(flag)

{

minimum\_time = 25;

for(n = 0; n < total\_frames; n++)

{

if(arr[frame[n]] == arr[k] && time[frame[n]] < minimum\_time)

{

temp = n;

minimum\_time = time[frame[n]];

}

}

arr[frame[temp]] = 0;

frame[temp] = pages[m];

}

for(n = 0; n < total\_frames; n++)

{

printf("%d\t", frame[n]);

}

printf("\n");

}

float faults;

faults=total\_pages-hit;

printf("\n\nPage Faults:\t%d\n",faults);

printf("Page Hit:\t%d\n",hit);

float hitratio;

float hitt=hit;

float pagess=total\_pages;

hitratio=hitt/pagess;

printf("Hit Ratio:\t%.2f\n",hitratio);

float missratio;

missratio=1-hitratio;

printf("Miss Ratio:\t%.2f\n\n\n",missratio);

return 0;

}

int main()

{

int ch;

do{

printf("1.LFU\n2.MFU\n3.Exit\n");

//printf("for mfu press '2'\n");

printf("Enter your choice: ");

scanf("%d",&ch);

switch(ch)

{

case 1:

lfu();

break;

case 2:

mfu();

break;

case 3:

exit(0);

break;

default:

printf("invalid choice");

break;

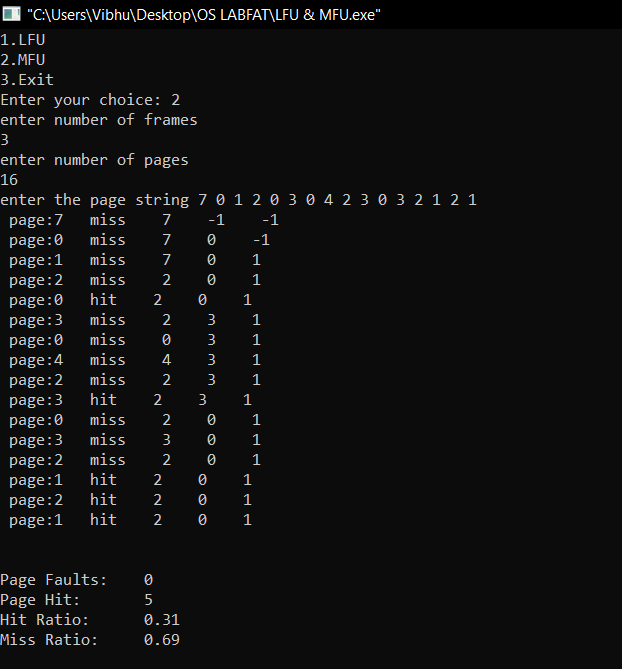
}

}while(1);

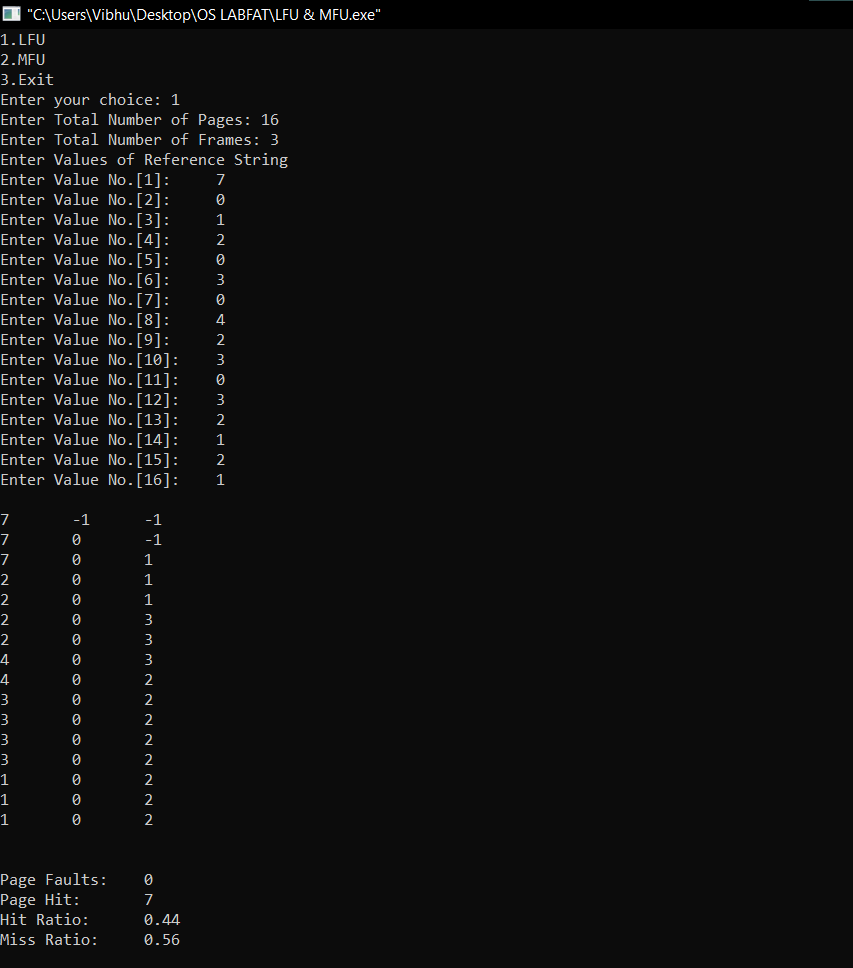
}

**Output(screenshots):**

**MFU:**



**LFU:**



**Q5b)** **Write a program to implement the following scenario using Bankers algorithm :**

**Determine if a deadlock situation exists for the following description of a resource**

**allocation graph, let the set of processes be P = { P1, P2, P3, P4}; let the set of**

**resource types be: R = { R1, R2, R3, R4, R5, R6}; Here, R1 has two instance of**

**resource type, R2 has one instance of resource type,R3 has two instance of resource**

**type, R4 has two instance of resource type, R5 has one instance of resource type, and**

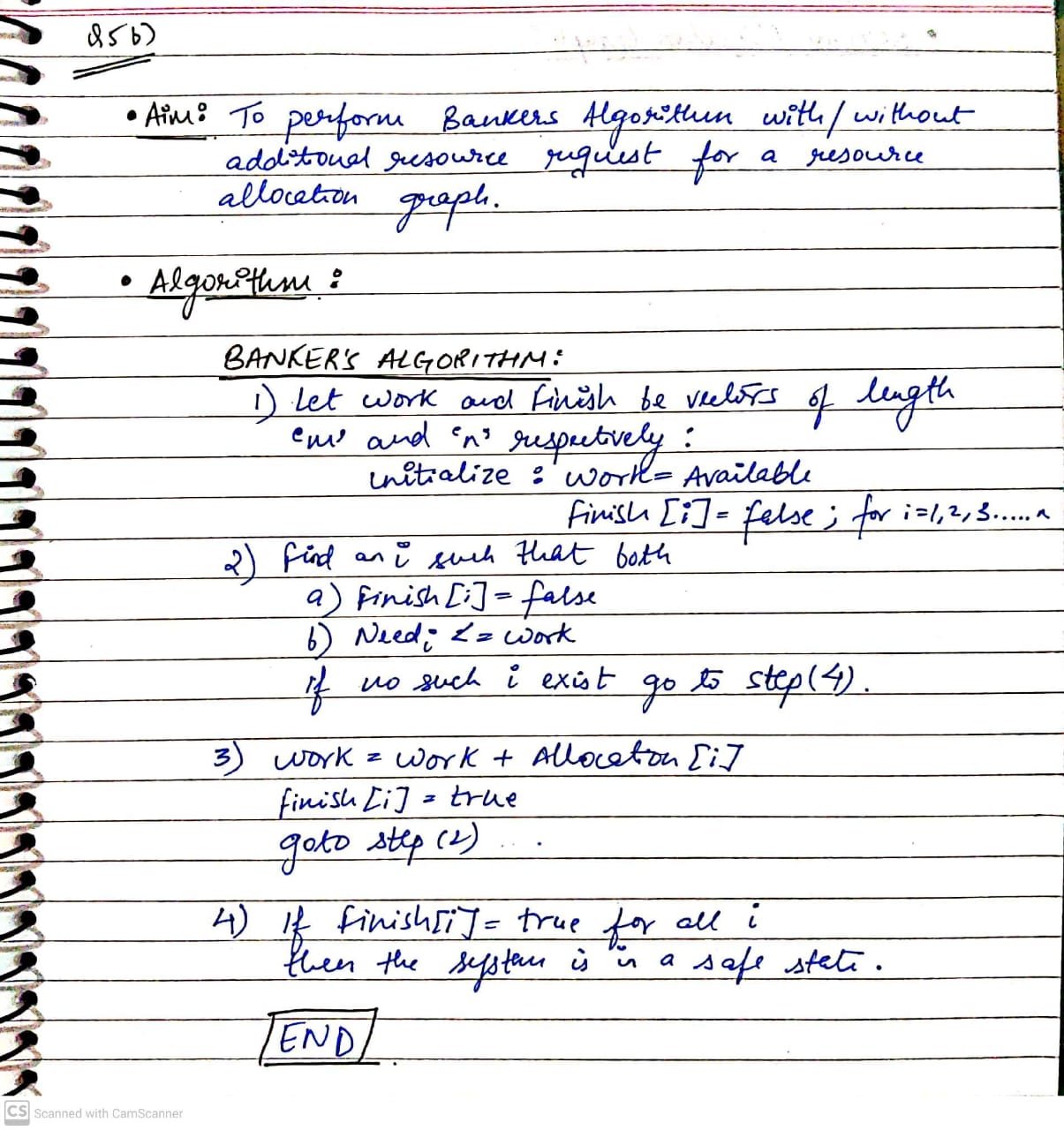
**R6 has two instance of resource type; let the set of request and assignment edges be:**

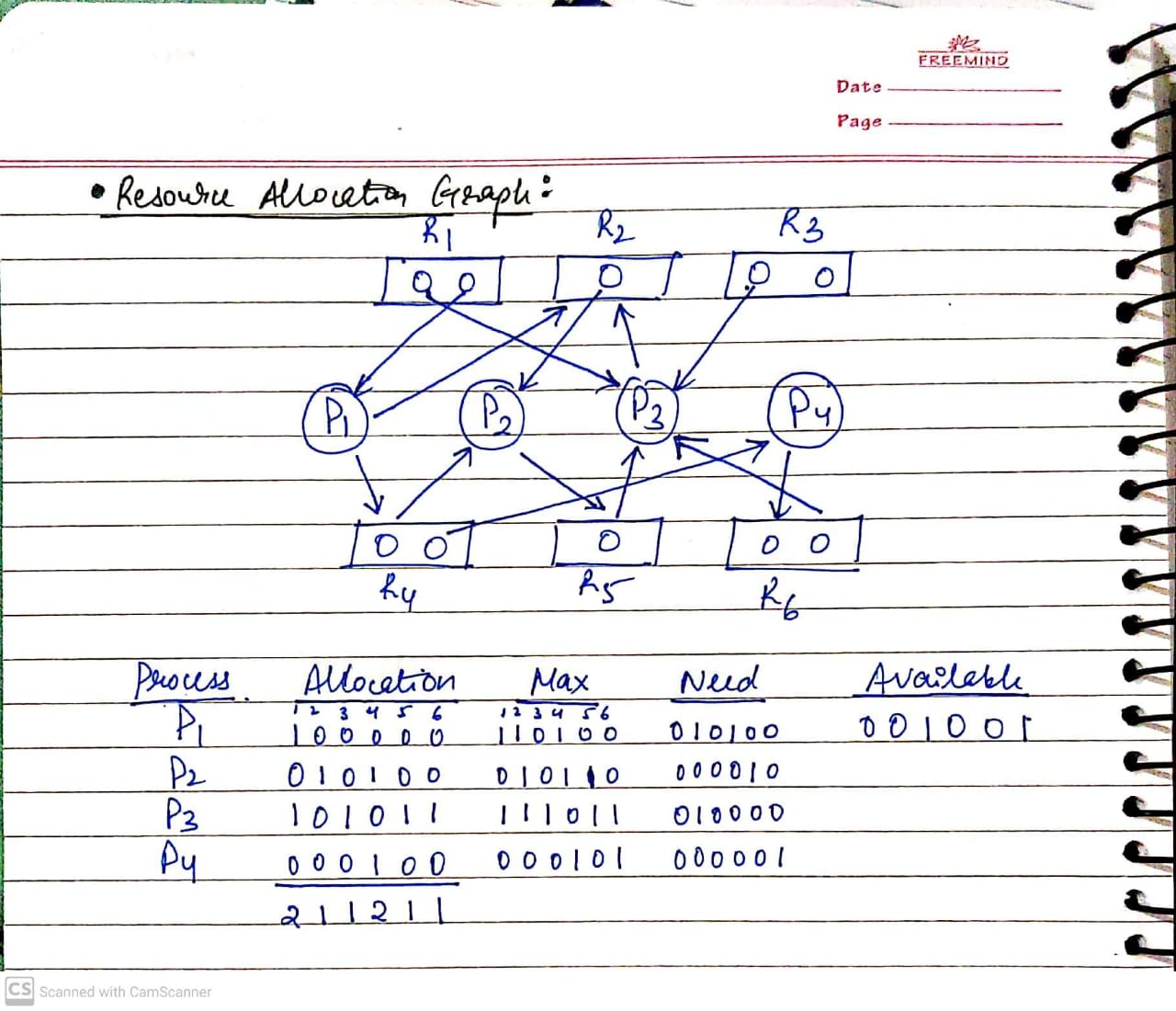
**E= { R1->P3, R1->P1, R2->P2, R3->P3,P1->R2, P1->R4, P4->R6, R4->P4, P3->R2,**

**R5->P3, P2->R5,R4->P2, R6->P3 }**

**A5b)**

**Handwritten file:**

****

****

**Code:**

#include<stdio.h>

#include<stdlib.h>

void print(int x[][10],int n,int m){

int i,j;

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

printf("\n");

for(j=0;j<m;j++){

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

}

}

}

//Resource Request algorithm

void res\_request(int A[10][10],int N[10][10],int AV[10][10],int pid,int m)

{

int reqmat[1][10];

int i;

printf("\n Enter additional request :- \n");

for(i=0;i<m;i++){

printf(" Request for resource %d : ",i+1);

scanf("%d",&reqmat[0][i]);

}

for(i=0;i<m;i++)

if(reqmat[0][i] > N[pid][i]){

printf("\n Error encountered.\n");

exit(0);

}

for(i=0;i<m;i++)

if(reqmat[0][i] > AV[0][i]){

printf("\n Resources unavailable.\n");

exit(0);

}

for(i=0;i<m;i++){

AV[0][i]-=reqmat[0][i];

A[pid][i]+=reqmat[0][i];

N[pid][i]-=reqmat[0][i];

}

}

//Safety algorithm

int safety(int A[][10],int N[][10],int AV[1][10],int n,int m,int a[]){

int i,j,k,x=0;

int F[10],W[1][10];

int pflag=0,flag=0;

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

F[i]=0;

for(i=0;i<m;i++)

W[0][i]=AV[0][i];

for(k=0;k<n;k++){

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

if(F[i] == 0){

flag=0;

for(j=0;j<m;j++){

if(N[i][j] > W[0][j])

flag=1;

}

if(flag == 0 && F[i] == 0){

for(j=0;j<m;j++)

W[0][j]+=A[i][j];

F[i]=1;

pflag++;

a[x++]=i;

}

}

}

if(pflag == n)

return 1;

}

return 0;

}

//Banker's Algorithm

void accept(int A[][10],int N[][10],int M[10][10],int W[1][10],int \*n,int \*m){

int i,j;

printf("\n Enter total no. of processes : ");

scanf("%d",n);

printf("\n Enter total no. of resources : ");

scanf("%d",m);

for(i=0;i<\*n;i++){

printf("\n Process %d\n",i+1);

for(j=0;j<\*m;j++){

printf(" Allocation for resource %d : ",j+1);

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

printf(" Maximum for resource %d : ",j+1);

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

}

}

printf("\n Available resources : \n");

for(i=0;i<\*m;i++){

printf(" Resource %d : ",i+1);

scanf("%d",&W[0][i]);

}

for(i=0;i<\*n;i++)

for(j=0;j<\*m;j++)

N[i][j]=M[i][j]-A[i][j];

printf("\n Allocation Matrix");

print(A,\*n,\*m);

printf("\n Maximum Requirement Matrix");

print(M,\*n,\*m);

printf("\n Need Matrix");

print(N,\*n,\*m);

}

int banker(int A[][10],int N[][10],int W[1][10],int n,int m){

int j,i,a[10];

j=safety(A,N,W,n,m,a);

if(j != 0 ){

printf("\n\n");

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

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

printf("\n A safety sequence has been detected.\n");

return 1;

}else{

printf("\n Deadlock has occured.\n");

return 0;

}

}

int main(){

int ret;

int A[10][10];

int M[10][10];

int N[10][10];

int W[1][10];

int n,m,pid,ch;

printf("\n DEADLOCK AVOIDANCE USING BANKER'S ALGORITHM\n");

accept(A,N,M,W,&n,&m);

ret=banker(A,N,W,n,m);

if(ret !=0 ){

printf("\n Do you want make an additional request ? (1=Yes|0=No)");

scanf("%d",&ch);

if(ch == 1){

printf("\n Enter process no. : ");

scanf("%d",&pid);

res\_request(A,N,W,pid-1,m);

ret=banker(A,N,W,n,m);

if(ret == 0 )

exit(0);

}

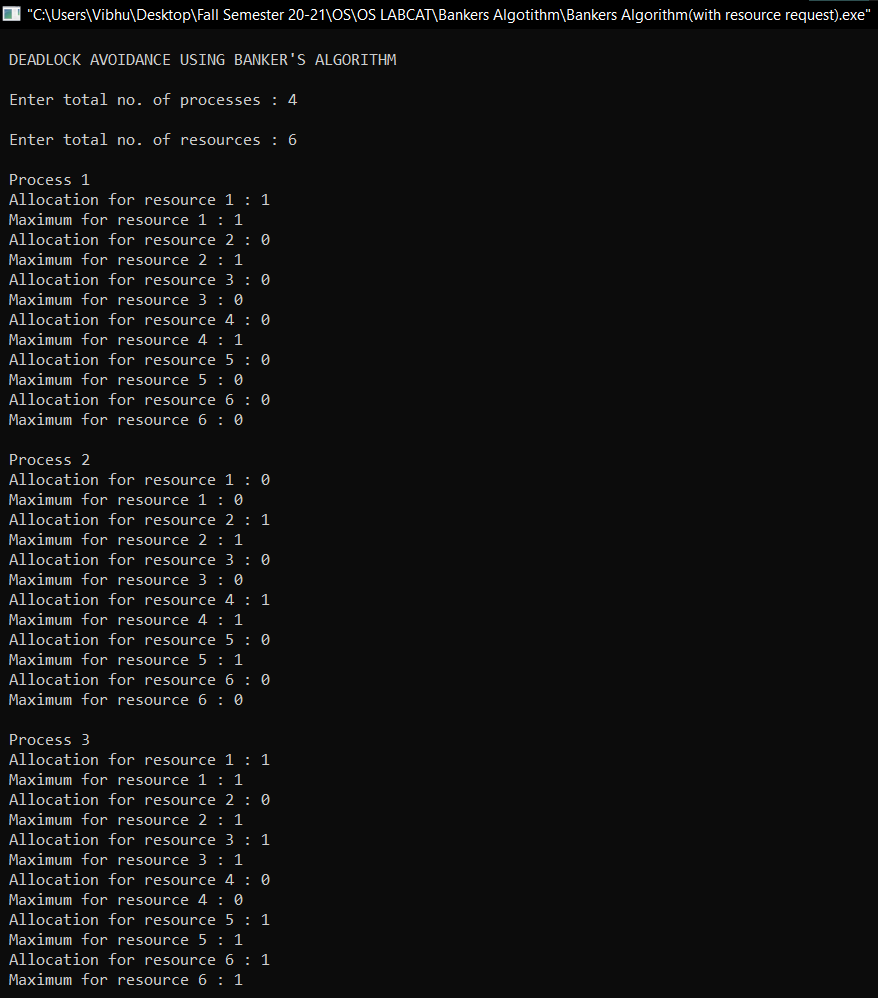
}else

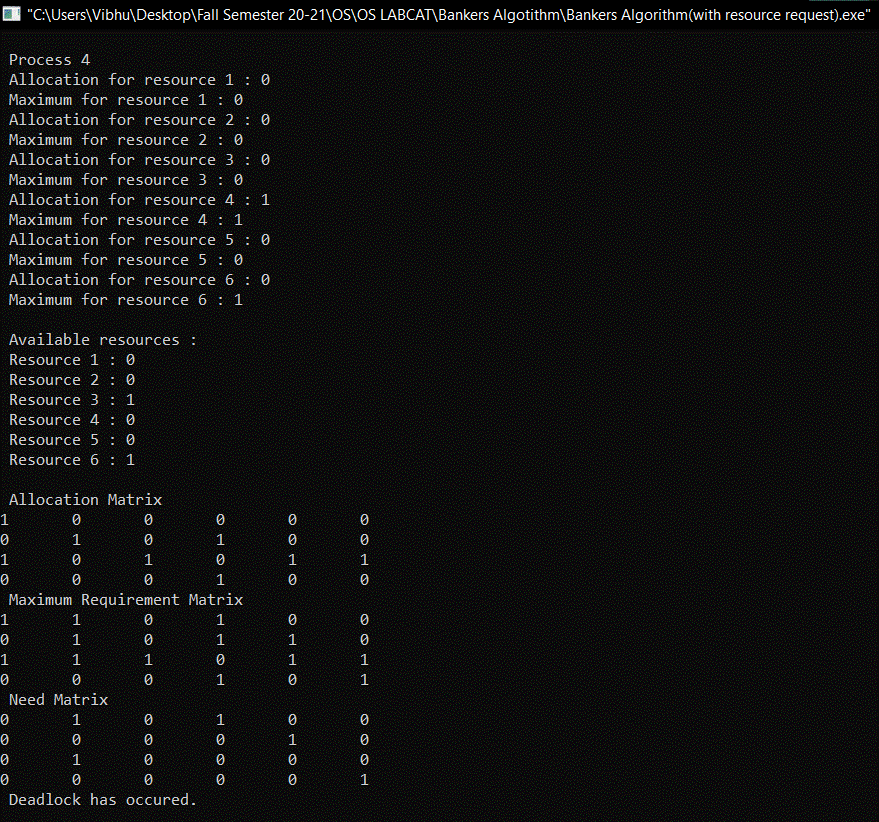
exit(0);

return 0;

}

**Output(screenshots):**





**INFERENCE: Deadlock will occur and there is no safety sequence detected.**