**Week - 6**

**(27 July 2023)**

**Experiment - 6**

**Question:**

1. Write a C program to simulate Banker’s Algorithm for the purpose of deadlock avoidance.
2. Write a C program to simulate deadlock detection.

**Program:**

1. **Banker’s Algorithm:**

#include <stdio.h>

int main()

{

int n, m, i, j, k;

printf("Enter the number of processes: ");

scanf("%d", &n);

printf("Enter the number of resources: ");

scanf("%d", &m);

int allocation[n][m];

printf("Enter the Allocation Matrix:\n");

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

{

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

{

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

}

}

int max[n][m];

printf("Enter the MAX Matrix:\n");

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

{

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

{

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

}

}

int available[m];

printf("Enter the Available Resources:\n");

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

{

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

}

int f[n], ans[n], ind = 0;

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

{

f[k] = 0;

}

int need[n][m];

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

{

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

{

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

}

}

int y = 0;

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

{

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

{

if (f[i] == 0)

{

int flag = 0;

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

{

if (need[i][j] > available[j])

{

flag = 1;

break;

}

}

if (flag == 0)

{

ans[ind++] = i;

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

{

available[y] += allocation[i][y];

}

f[i] = 1;

}

}

}

}

int flag = 1;

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

{

if (f[i] == 0)

{

flag = 0;

printf("The following system is not safe\n");

break;

}

}

if (flag == 1)

{

printf("Following is the SAFE Sequence\n");

for (i = 0; i < n - 1; i++)

{

printf(" P%d ->", ans[i]);

}

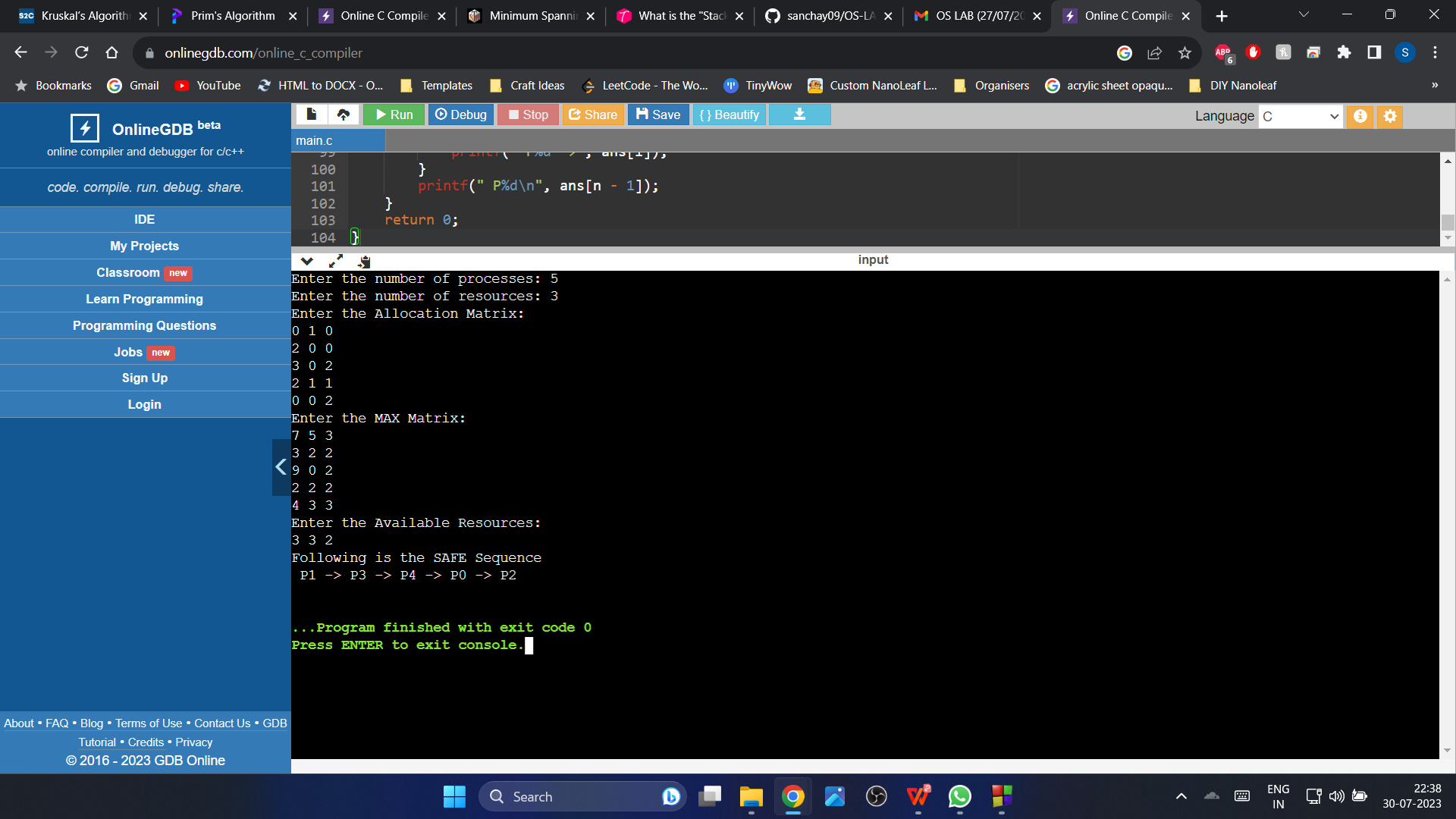
printf(" P%d\n", ans[n - 1]);

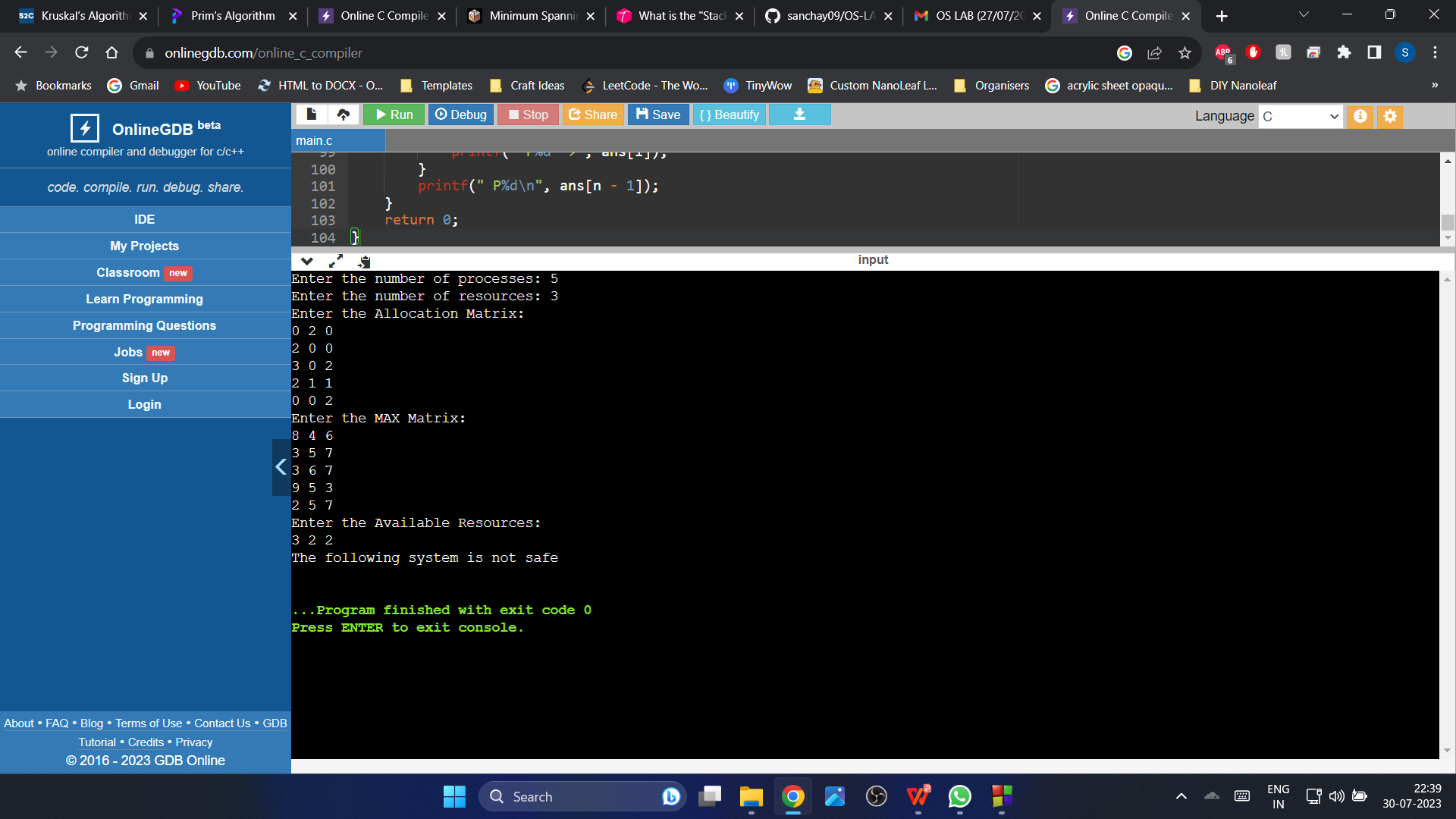
}

return 0;

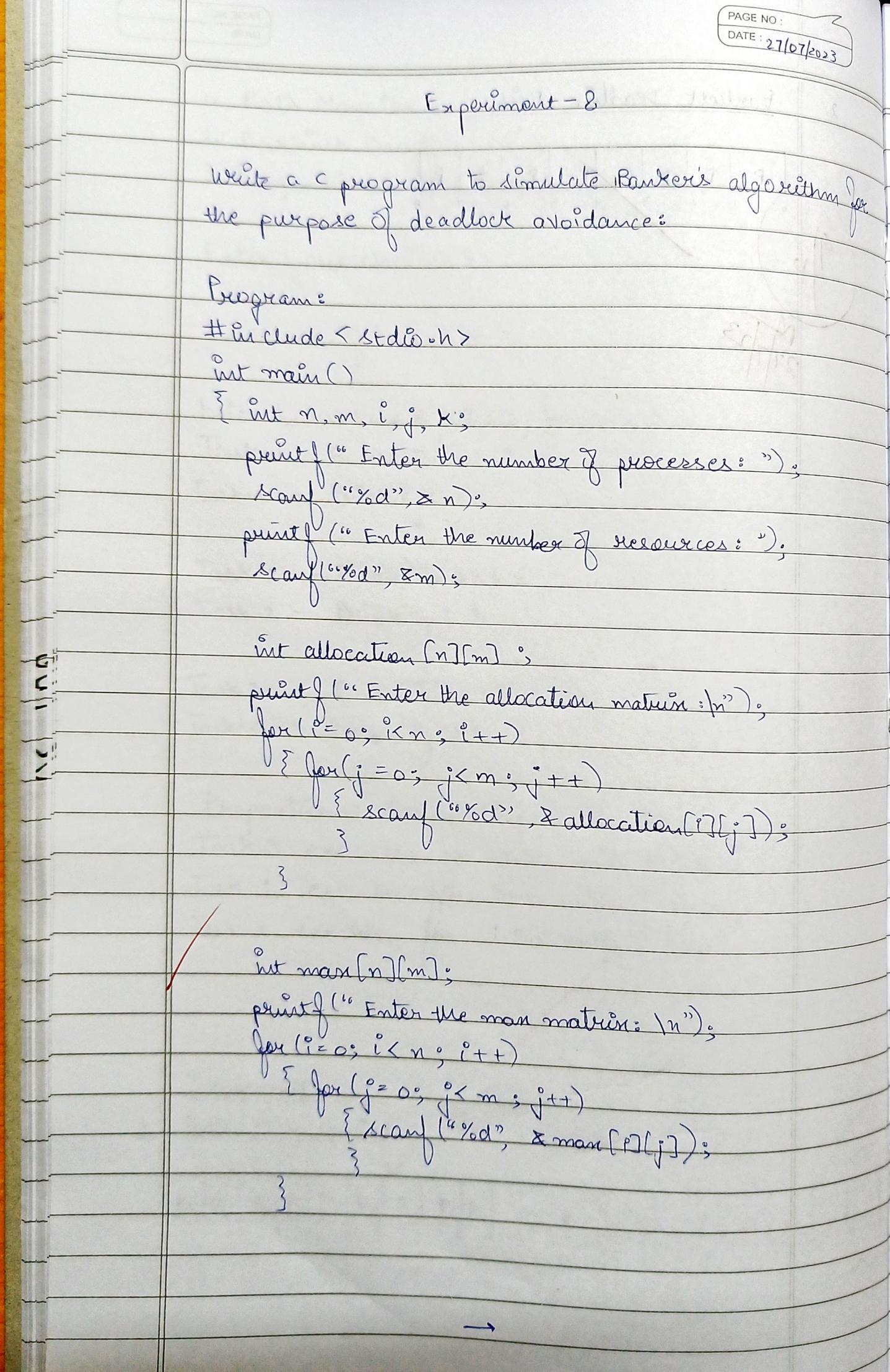
}

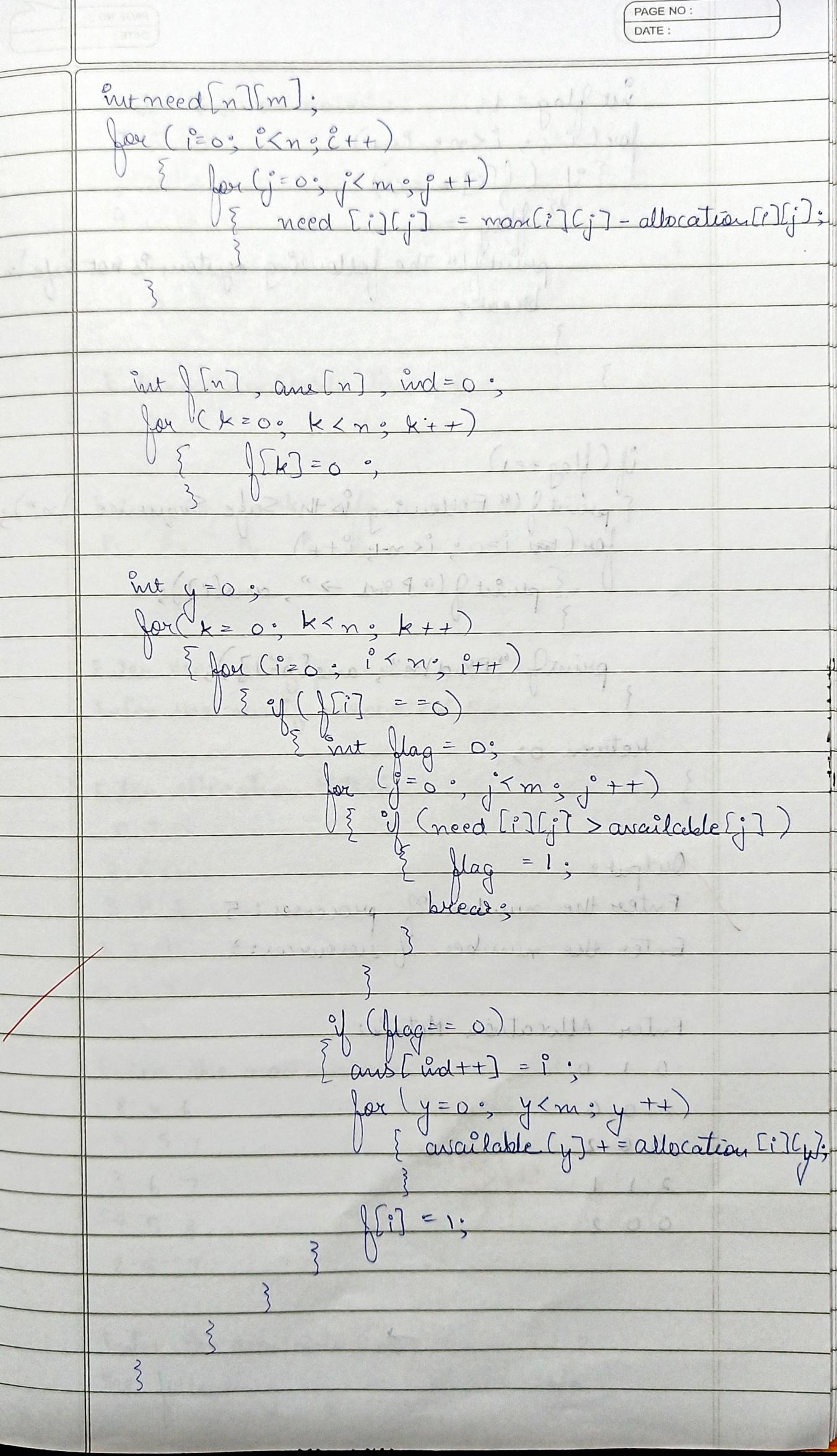
**Output: (Banker’s Algorithm)**

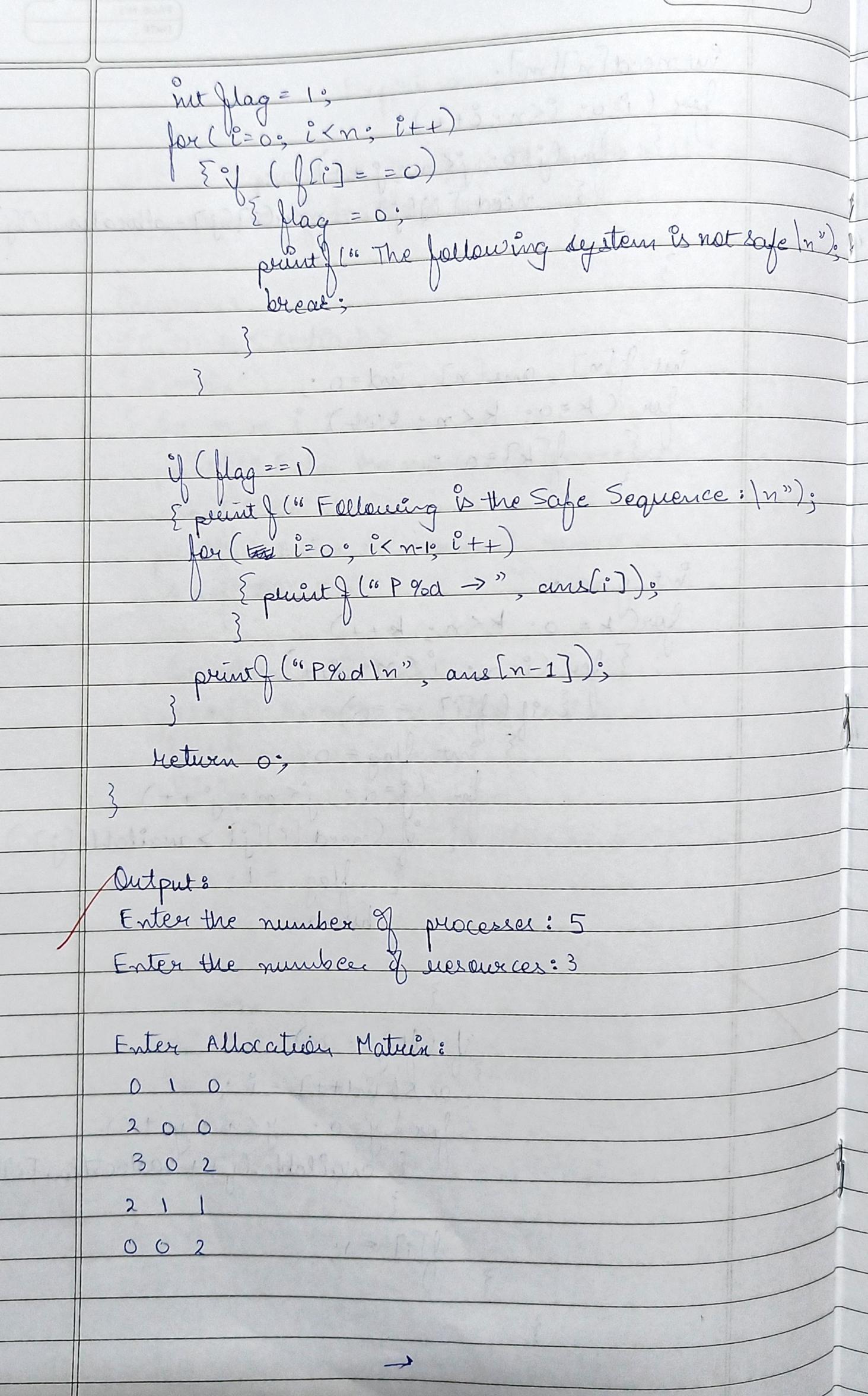


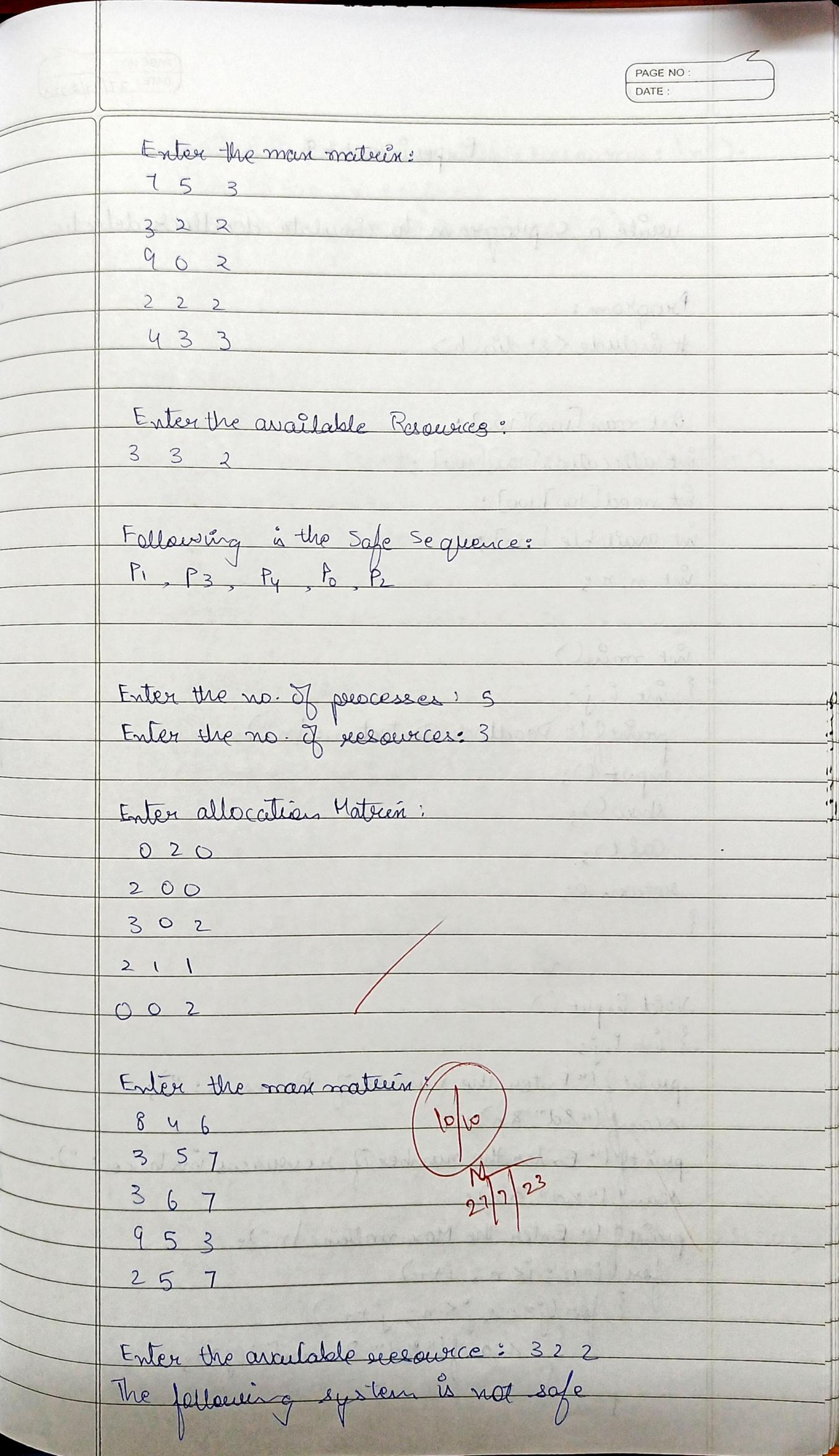


**Observation Book Pictures: (Banker’s Algorithm)**









1. **Deadlock Detection:**

**Program:**

#include<stdio.h>

int max[100][100];

int allocation[100][100];

int need[100][100];

int available[100];

int n,r;

int main()

{

int i,j;

printf("Deadlock Detection\n");

input();

show();

cal();

return 0;

}

void input()

{

int i,j;

printf("Enter the no of Processes: ");

scanf("%d",&n);

printf("Enter the no of resource instances: ");

scanf("%d",&r);

printf("Enter the Max Matrix:\n");

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

{

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

{

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

}

}

printf("Enter the Allocation Matrix:\n");

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

{

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

{

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

}

}

printf("Enter the available Resources:\n");

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

{

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

}

}

void show()

{

int i,j;

printf("Process\t Allocation\t Max\t Available\t");

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

{

printf("\nP%d\t ",i+1);

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

{

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

}

printf("\t");

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

{

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

}

printf("\t");

if(i==0)

{

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

printf("%d ",available[j]);

}

}

}

void cal()

{

int finish[100],temp,need[100][100],flag=1,k,c1=0;

int dead[100];

int safe[100];

int i,j;

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

{

finish[i]=0;

}

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

{

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

{

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

}

}

while(flag)

{

flag=0;

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

{

int c=0;

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

{

if((finish[i]==0)&&(need[i][j]<=available[j]))

{

c++;

if(c==r)

{

for(k=0;k<r;k++)

{

available[k]+=allocation[i][j];

finish[i]=1;

flag=1;

}

if(finish[i]==1)

{

i=n;

}

}

}

}

}

}

j=0;

flag=0;

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

{

if(finish[i]==0)

{

dead[j]=i;

j++;

flag=1;

}

}

if(flag==1)

{

printf("\n\nSystem is in Deadlock and the Deadlock process are\n");

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

{

printf("P%d\t",dead[i]);

}

}

else

{

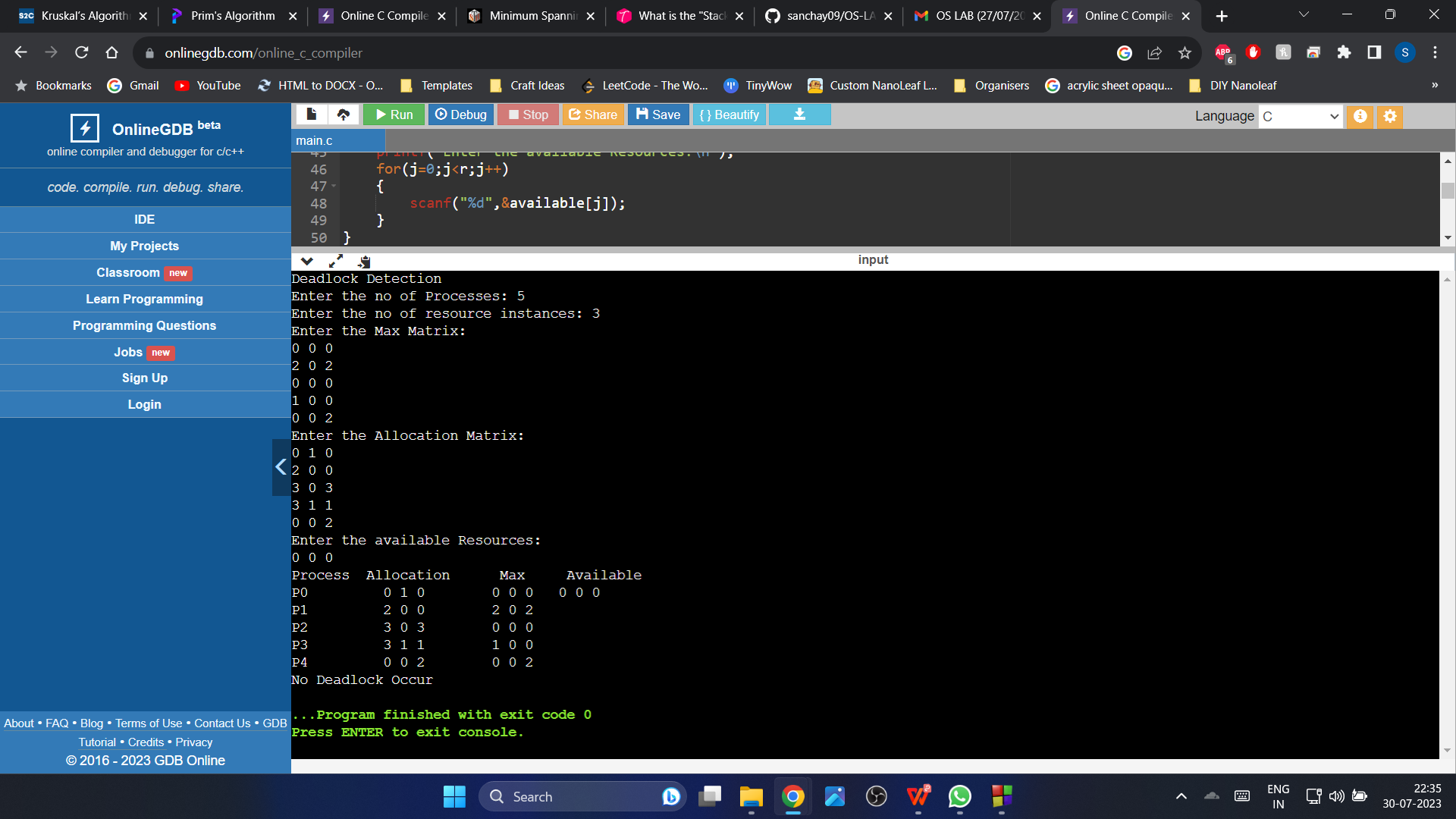
printf("\nNo Deadlock Occur");

}

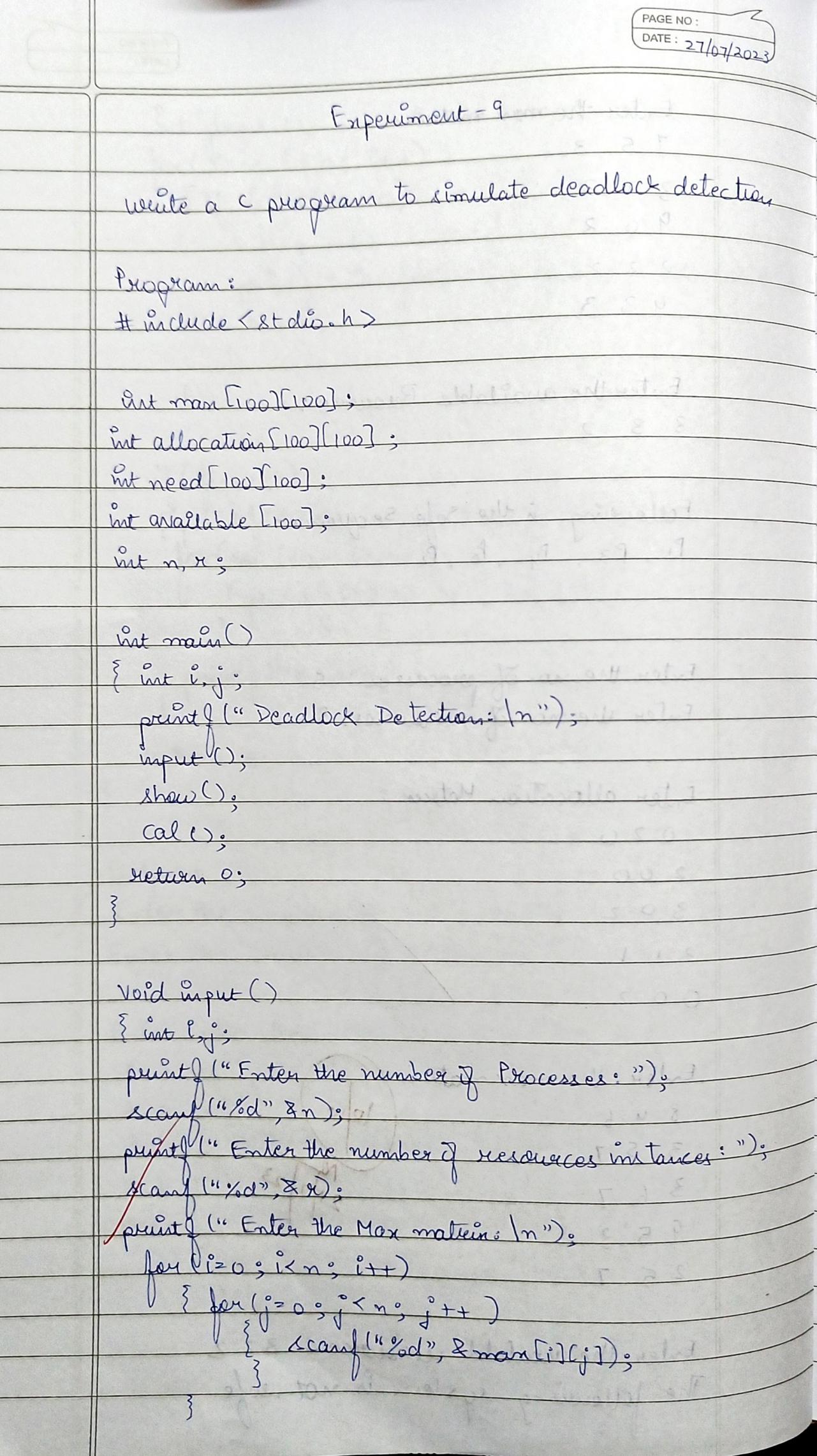
}

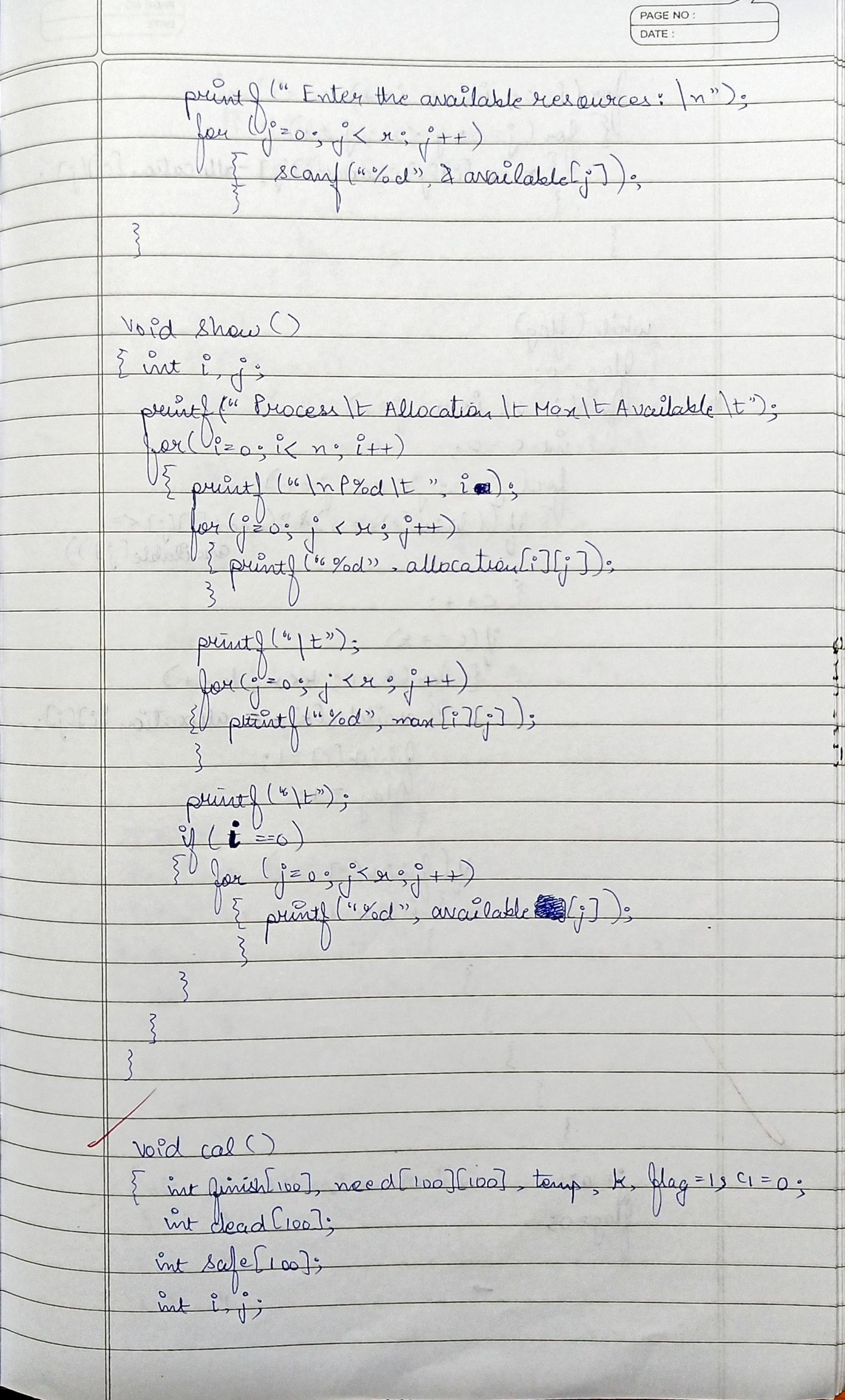
**Output: (Deadlock Detection)**

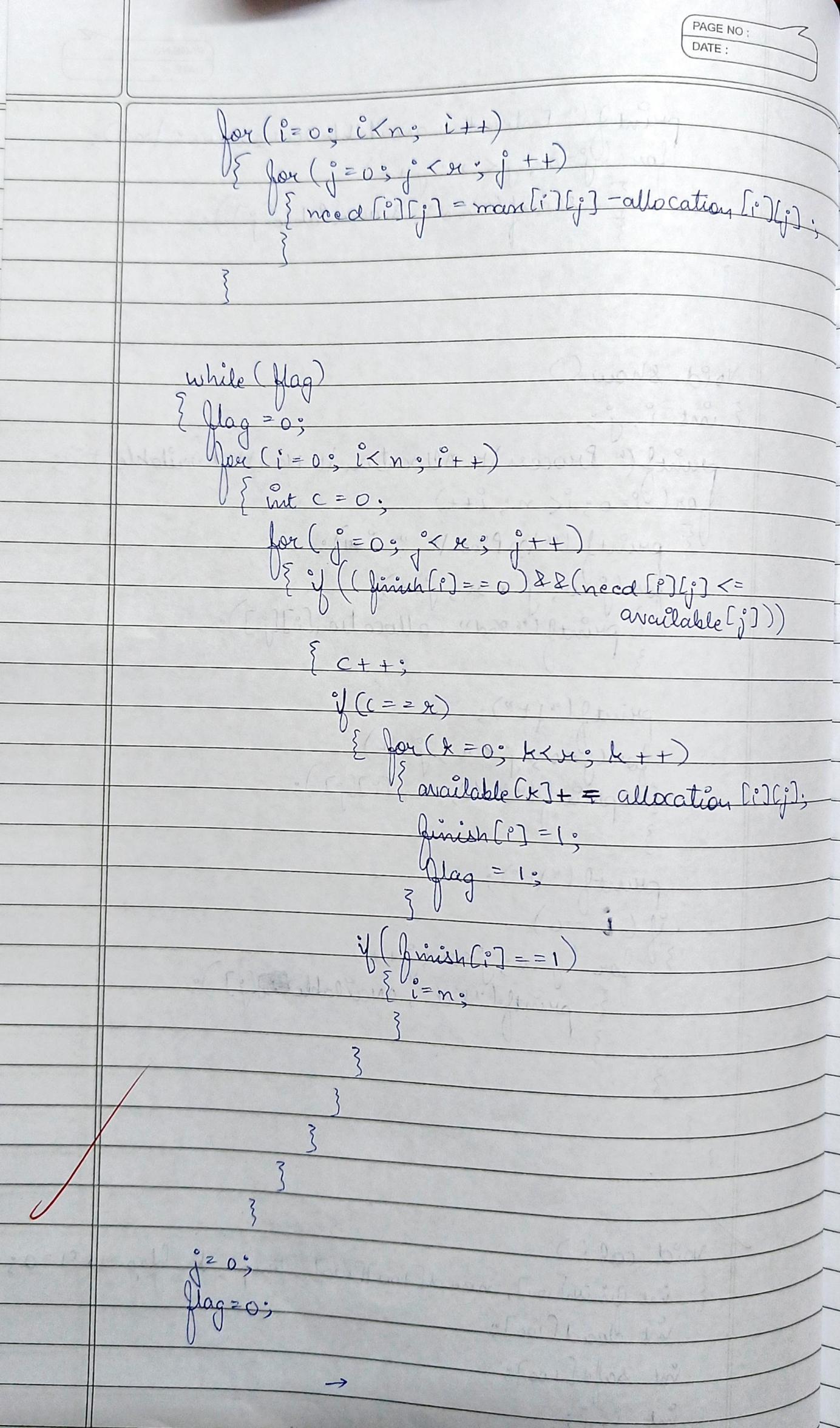
****

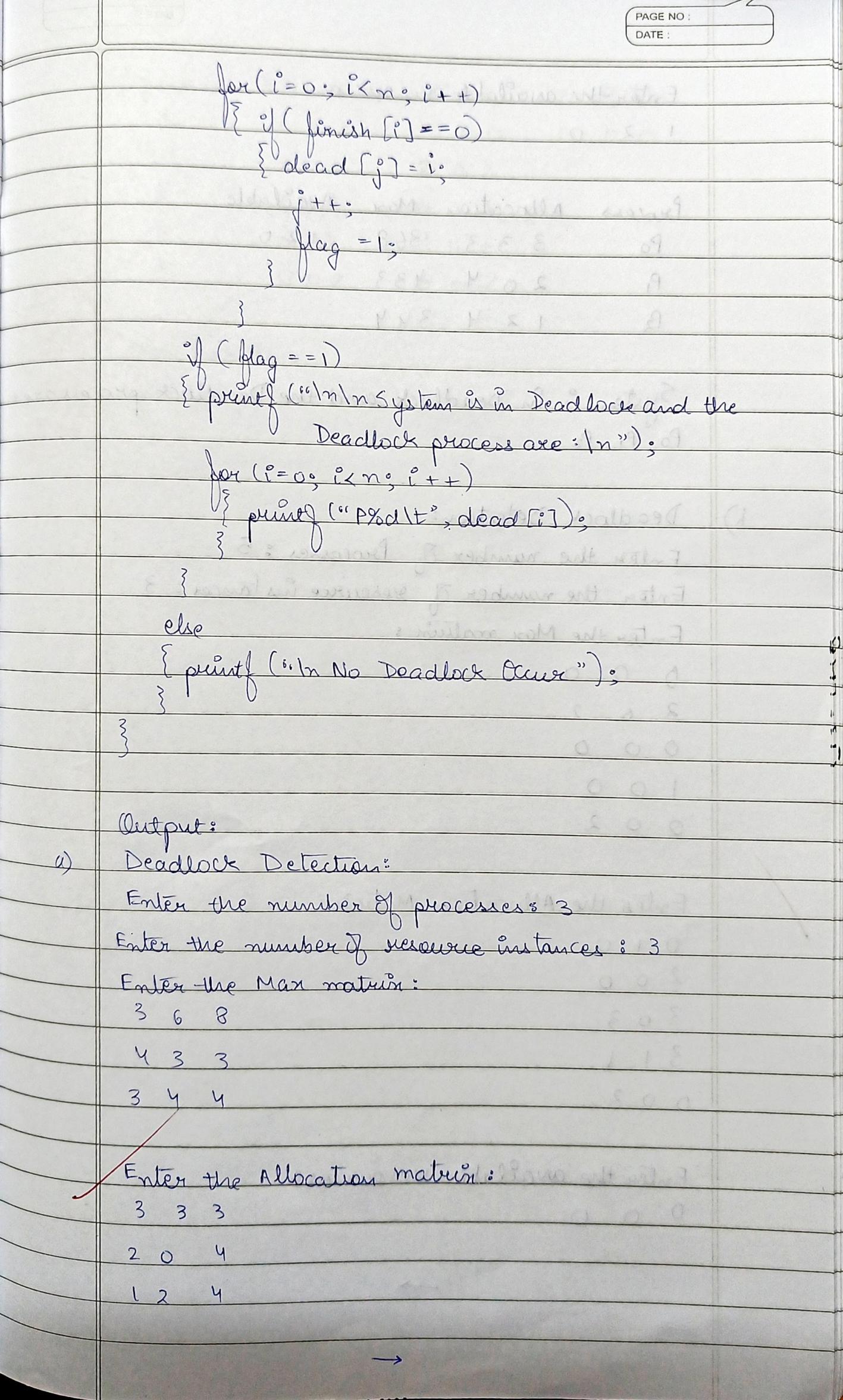
****

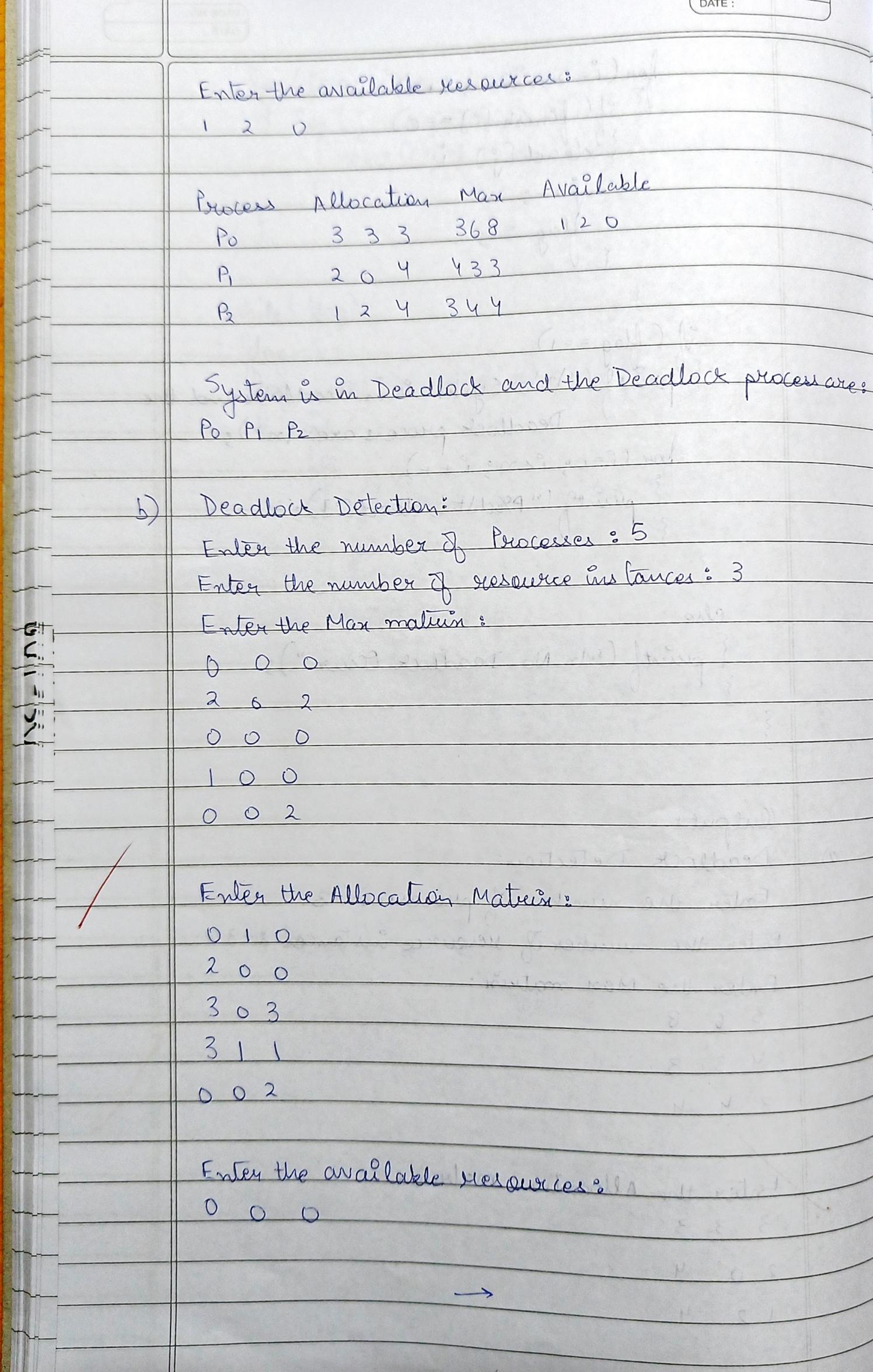
**Observation Book Pictures: (Deadlock Detection)**

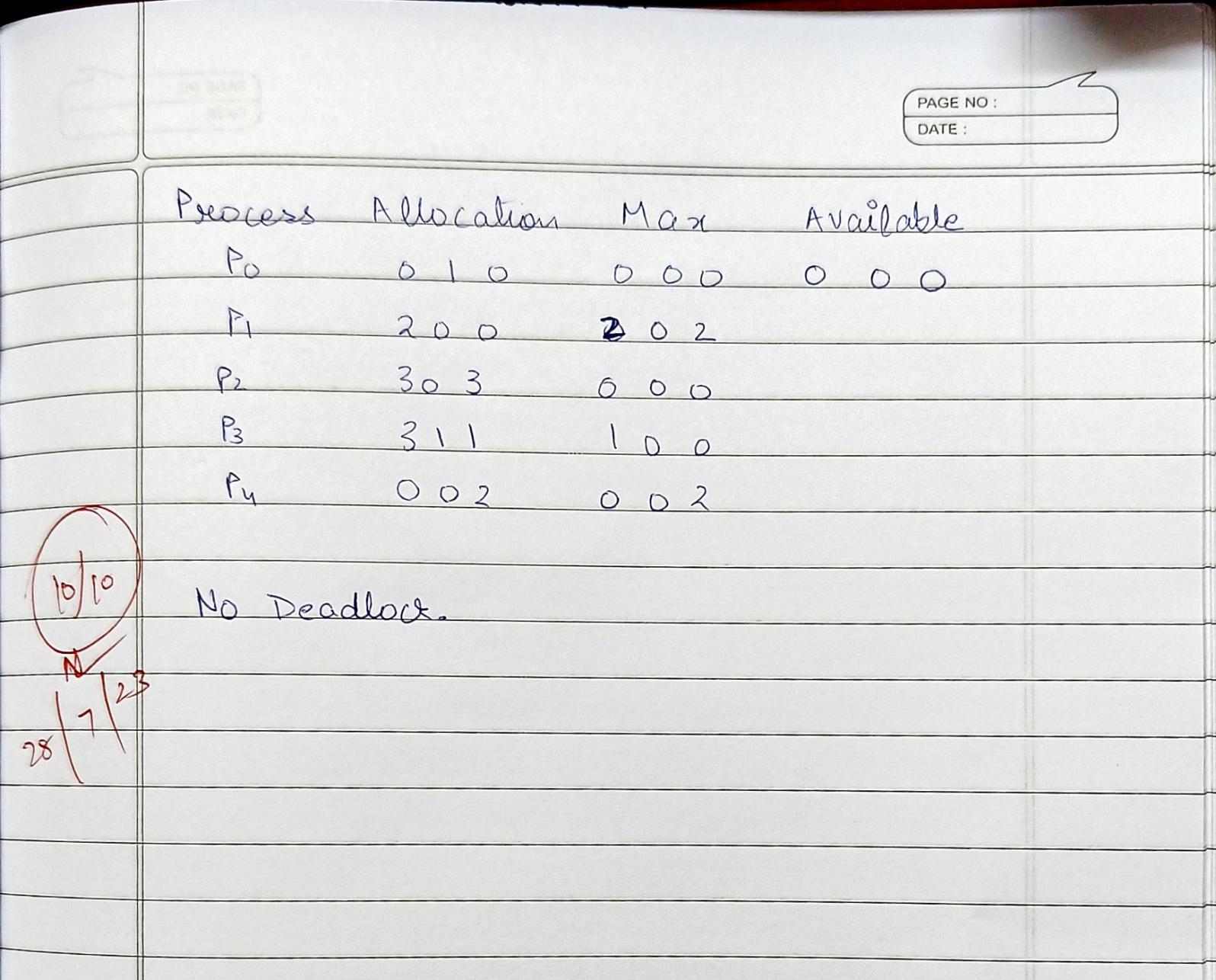
****

****

****

****

****

****