7.

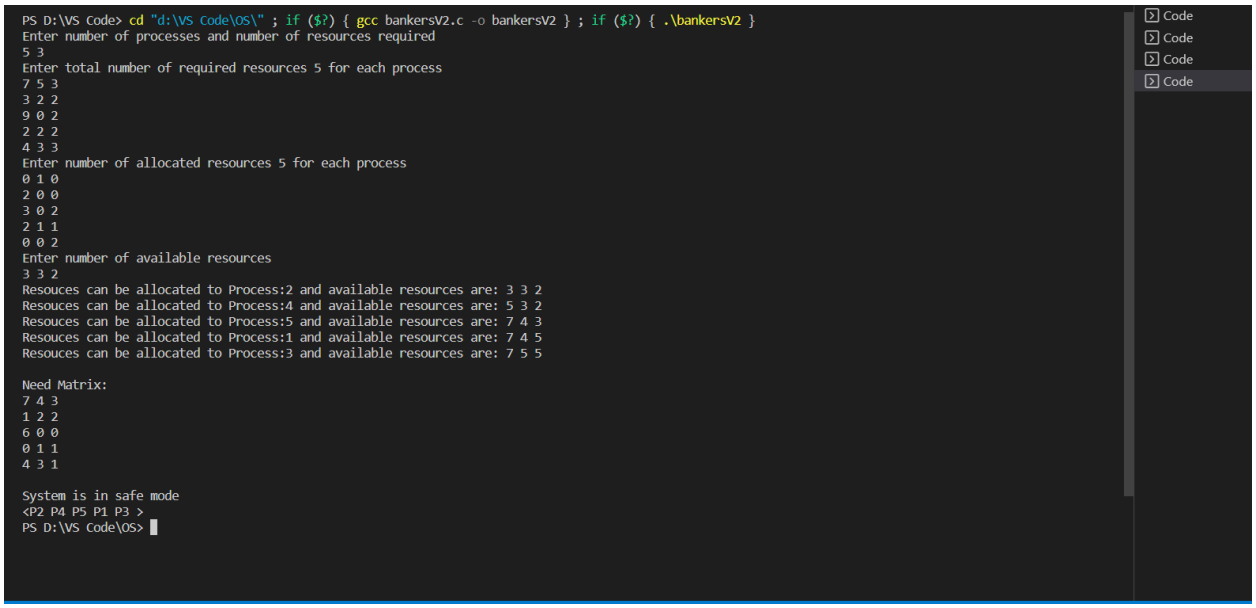
Code:

```
#include <stdio.h>

int main() {
   int n, m, all[10][10], req[10][10], ava[10], need[10][10];
   int i, j, k, flag[10], prev[10], c, count = 0,array[10],z=0;

   printf("Enter number of processes and number of resources required \n");
   scanf("%d %d", &n, &m);

   printf("Enter total number of required resources %d for each process\n", n);
   for (i = 0; i < n; i++)
      for (j = 0; j < m; j++)
         scanf("%d", &req[i][j]);

   printf("Enter number of allocated resources %d for each process\n", n);
   for (i = 0; i < n; i++)
      for (j = 0; j < m; j++)
         scanf("%d", &all[i][j]);

   printf("Enter number of available resources \n");
   for (i = 0; i < m; i++)
      scanf("%d", &ava[i]);

   for (i = 0; i < n; i++)
      for (j = 0; j < m; j++)
         need[i][j] = req[i][j] - all[i][j];

   for (i = 0; i < n; i++)
      flag[i] = 1;

   k = 1;

   while (k) {
      k = 0; // Reset the value of k for each iteration of the loop

      for (i = 0; i < n; i++) {
         if (flag[i]) {
            c = 0;
            for (j = 0; j < m; j++) {
               if (need[i][j] <= ava[j]) {
                  c++;
```

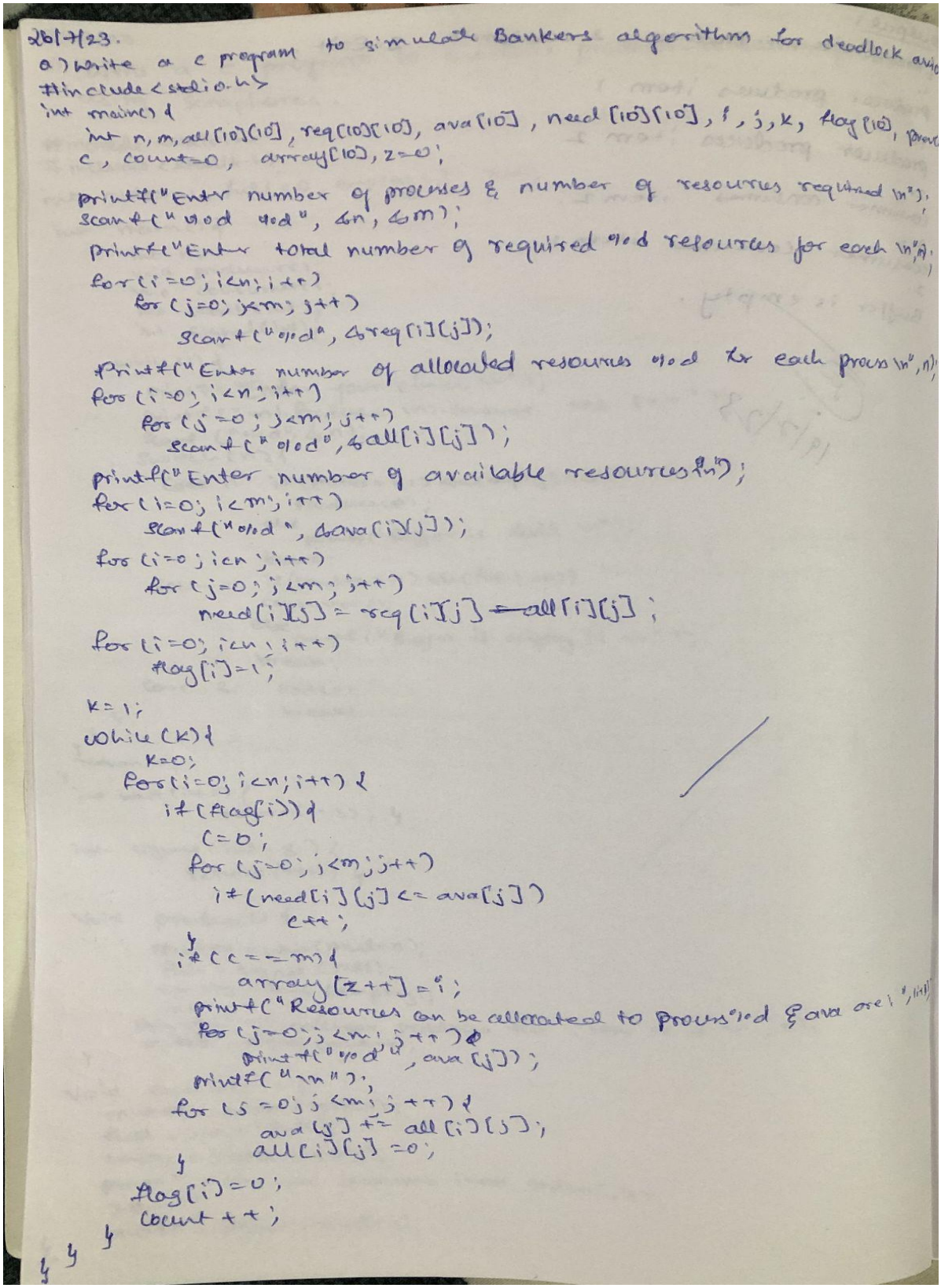
```
                }
            }
            if (c == m) {
                array[z++]=i;
                printf("Resouces can be allocated to Process:%d and available resources are: ", (i
+ 1));
                for (j = 0; j < m; j++) {
                    printf("%d ", ava[j]);
                }
                printf("\n");

                for (j = 0; j < m; j++) {
                    ava[j] += all[i][j];
                    all[i][j] = 0;
                }

                flag[i] = 0;
                count++;
            }
        }
    }

    // Check if the current state is different from the previous state
    for (i = 0; i < n; i++) {
        if (flag[i] != prev[i]) {
            k = 1;
            break;
        }
    }

    for (i = 0; i < n; i++) {
        prev[i] = flag[i];
    }
}

printf("\nNeed Matrix:\n");
for (i = 0; i < n; i++) //printing need matrix
{
    for (j = 0; j < m; j++)
        printf("%d ",need[i][j]);
    printf("\n");
}
```

```
    if (count == n) {
        printf("\nSystem is in safe mode \n<");
        for(i=0;i<n;i++)
        printf("P%d ",(array[i]+1));
        printf(">\n");
    } else {
        printf("\nSystem is not in safe mode deadlock occurred \n");
    }
    return 0;
}
```

Output:



Observation:



26/7/23.

a) Write a c program to simulate Bankers algorithm for deadlock avoi

```
#include<stdio.h>
int main(){
    int n,m,all[10][10],req[10][10],ava[10],need[10][10],i,j,k,flag[10],prev
    c, count=0, array[10],z=0;

    printf("Enter number of process & number of resources required \n");
    scanf("%d %d",&n,&m);
    printf("Enter total number of required %d resources for each \n",n);
    for(i=0;i<n;i++)
        for(j=0;j<m;j++)
            scanf("%d",&req[i][j]);
    printf("Enter number of allocated resources %d for each process \n",n);
    for(i=0;i<n;i++)
        for(j=0;j<m;j++)
            scanf("%d",&all[i][j]);
    printf("Enter number of available resources\n");
    for(i=0;i<m;i++)
        scanf("%d",&ava[i][j]);
    for(i=0;i<n;i++)
        for(j=0;j<m;j++)
            need[i][j]=req[i][j]-all[i][j];
    for(i=0;i<n;i++)
        flag[i]=1;

    k=1;
    while(k){
        k=0;
        for(i=0;i<n;i++){
            if(flag[i]){
                c=0;
                for(j=0;j<m;j++)
                    if(need[i][j]<=ava[j])
                        c++;
                if(c==m){
                    array[z++]=i;
                    printf("Resources can be allocated to process %d & ava are: ",i);
                    for(j=0;j<m;j++){
                        printf("%d",ava[j]);
                    printf("\n");
                    for(j=0;j<m;j++){
                        ava[j]+=all[i][j];
                        all[i][j]=0;
                    }
                flag[i]=0;
                count++;
            }
        }
    }
}
```

```
for(i=0; i<n; i++){
        if(flag[i] != prev[i]){
            k=1;
            break;
        }
    }
    for(i=0; i<n; i++)
        prev[i] = flag[i];
}
printf("\nNeed Matrix :\n");
    for(i=0; i<n; i++)
        for(j=0; j<m; j++)
            printf("%d ", need[i][j]);
if(count==n){
        printf("\nSystem is in Safe mode\n");
        for(i=0; i<n; i++)
            printf("p%d ", (arr[i]+1));
        printf("\n");
    }else
        printf("\nSystem is not in safe mode deadlock occured\n");
return 0;
}
```

Output :

```
Enter number of process & no. of resources  5 3
Enter total number of required resources
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
Enter allocated resources.
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
Enter number of available resources.  3 3 2.
Resources can be allocated to Process: 2 & available are: 3 3 2
Resources can be allocated to process 4 & available are: 5 3 2
Resources can be allocated to process: 5 & available are: 7 4 3
Resources can be allocated to process : 1 & available are: 7 4 5
Resources can be allocated to process: 3 & available are: 7 5 5

Need matrix:
7 4 3
1 2 2
6 0 0
0 1 1
4 3 1
```

26/7