8. Write a C program to simulate deadlock detection

Code:

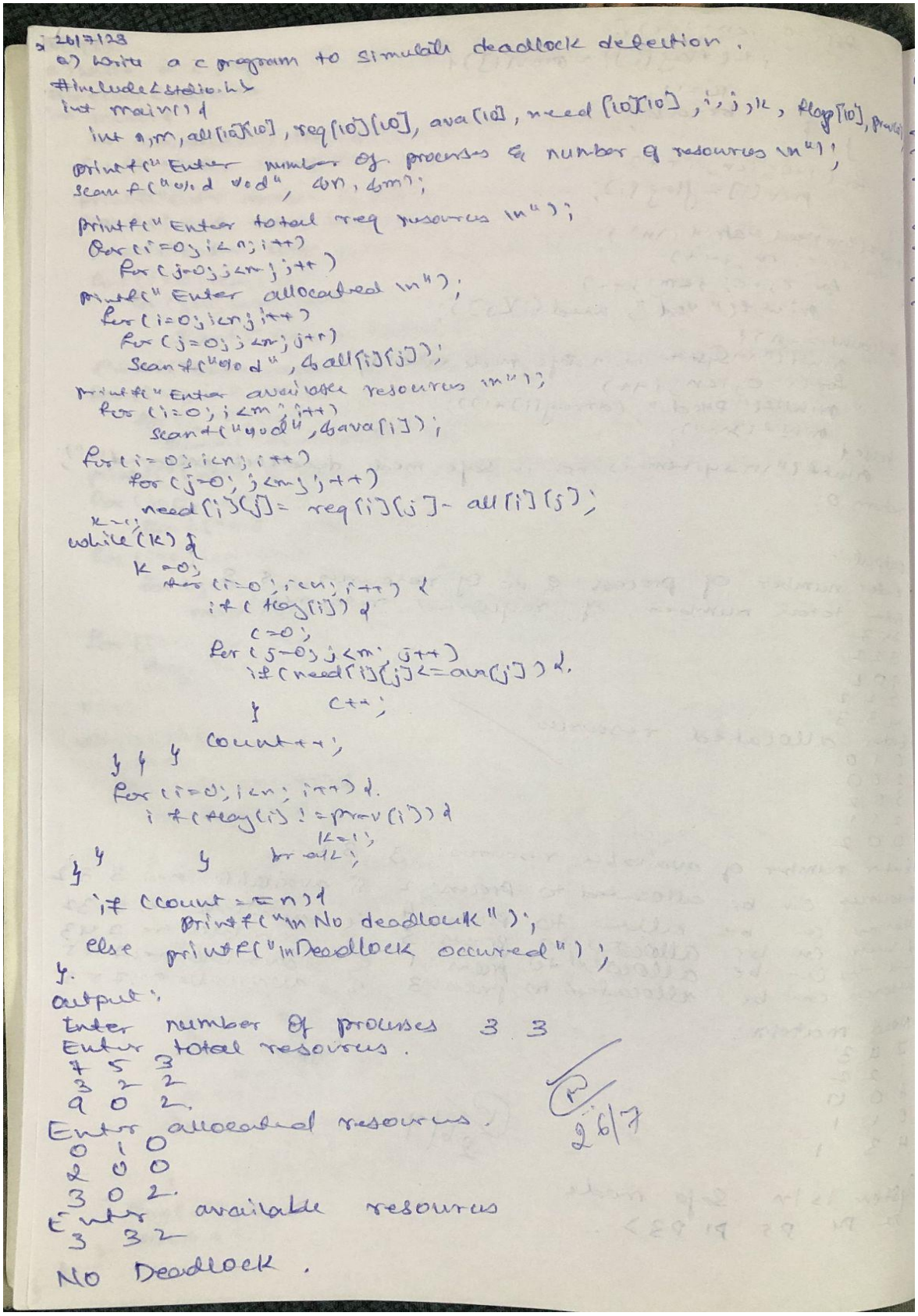
```
#include <stdio.h>

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;

    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) {
                for (j = 0; j < m; j++) {
                }
                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];
    }
  }
  if (count == n) {
    printf("\nNo deadlock");
  } else {
    printf("\nDeadlock occurred \n");
  }
  return 0;
}
```

Output:

```
PS D:\VS Code\OS> cd "d:\VS Code\OS\" ; if ($?) { gcc bankersV2.c -o bankersV2 } ; if ($?) { .\bankersV2 }
Enter number of processes and number of resources required
3 3
Enter total number of required resources 3 for each process
7 5 3
3 2 2
9 0 2
Enter number of allocated resources 3 for each process
0 1 0
2 0 0
3 0 2
Enter number of available resources
1 2 3
```

Observation:



2017128

a) Write a c program to simulate deadlock detection.

```
#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[10];
  printf("Enter number of process & number of resources \n");
  scanf("%d %d", &n, &m);
  printf("Enter total req resources \n");
  for(i=0;i<n;i++)
    for(j=0;j<m;j++)
  printf("Enter allocated \n");
    for(i=0;i<n;i++)
      for(j=0;j<m;j++)
      scanf("%d", &all[i][j]);
  printf("Enter 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];
  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++;
            }
          count++;
  }}}
  for(i=0;i<n;i++){
    if(flag[i]!=prev[i]){
      k=1;
      break;
  }}
  if(count==n){
      printf("\n No deadlock");
  else printf("\nDeadlock occured");
}
```

Output:

```
Enter number of proucses  3 3
Enter total resources.
7 5 3
3 2 2
9 0 2.
Enter allocated resources.
0 1 0
2 0 0
3 0 2.
Enter available resources
3 3 2
NO Deadlock.
```

26/7