# WEEK 7

Write a C program to simulate Bankers algorithm for the purpose of deadlock avoidance.

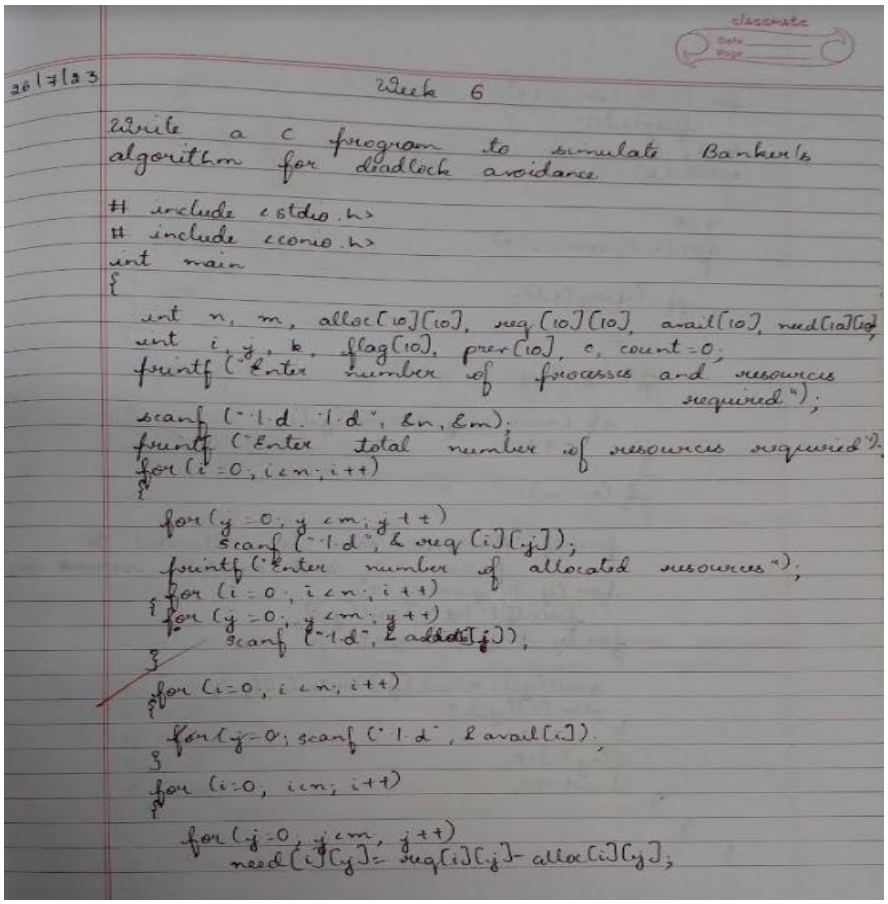
CODE :

```
#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;
        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) {
                    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++;
            }
        }
    }
        if (flag[i] != prev[i]) {
            k = 1;
            break;
        }
    }

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

  if (count == n) {
    printf("\nSystem is in safe mode ");
  } else {
    printf("\nSystem is not in safe mode deadlock occurred \n");
  }
  return 0;
}
```

OBSERVATION :

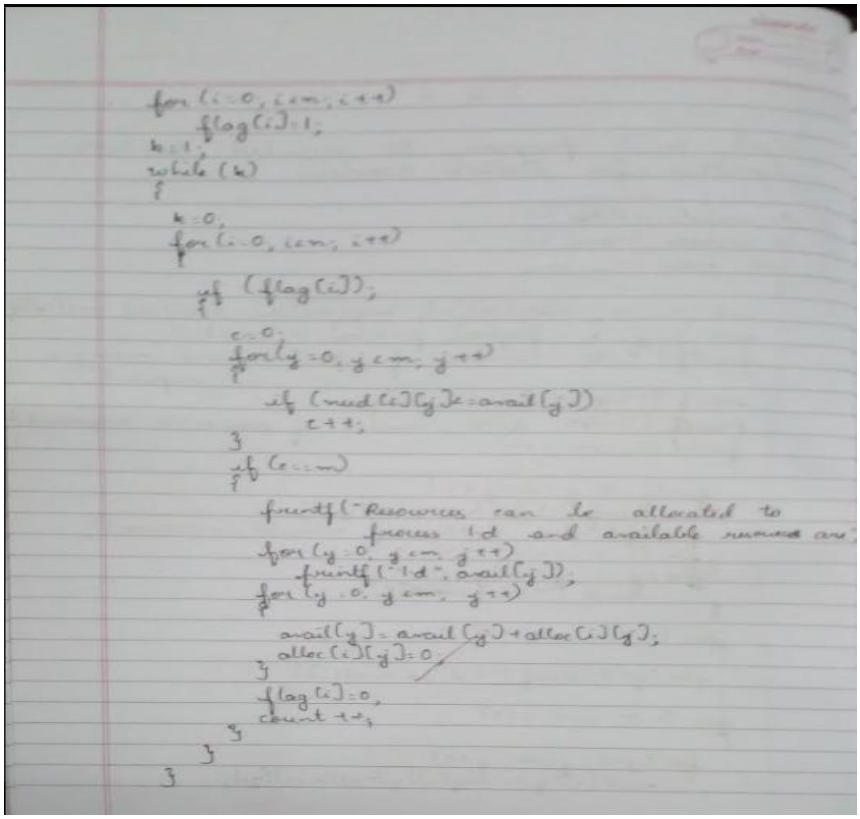


20/7/23

Week 6

Write a C program to simulate Banker's algorithm for deadlock avoidance

```
#include <stdio.h>
#include <conio.h>
int main
{
  int n, m, alloc[10][10], req[10][10], avail[10], need[10][10];
  int i, j, k, flag[10], prev[10], c, count=0;
  printf("Enter number of processes and resources required");
  scanf("%d %d", &n, &m);
  printf("Enter total number of resources required");
  for(i=0; i<n; i++)
  {
    for(j=0; j<m; j++)
      scanf("%d", &req[i][j]);
    printf("Enter number of allocated resources");
    for(i=0; i<n; i++)
    for(j=0; j<m; j++)
      scanf("%d", &alloc[i][j]);
  }
  for(i=0; i<n; i++)
  {
    for(j=0; scanf("%d", &avail[i]);
  }
  for(i=0; i<n; i++)
  {
    for(j=0; j<m; j++)
      need[i][j]= req[i][j]- alloc[i][j];
```



```
  for(i=0; i<m; 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]<=avail[j])
            c++;
        }
        if(c==m)
        {
          printf("Resources can be allocated to
                  process %d and available resources are",i);
          for(j=0; j<m; j++)
            printf("%d", avail[j]);
          for(j=0; j<m; j++)
          {
            avail[j]= avail[j]+alloc[i][j];
            alloc[i][j]=0;
          }
          flag[i]=0;
          count++;
        }
      }
    }
  }
```

```
    for(i=0;i<n;i++)
      if(flag[i]!=prev[i])
        c=1;
        break;
    }
    for(i=0;i<n;i++)
      prev[i]=flag[i];
  }
  for(i=0;i<n;i++)
  {
    for(j=0;j<m;j++)
      printf("%d ",need[i][j]);
  }
  if(count==n)
    printf("System is safe");
  else
    printf("System is not safe, deadlock occurred");
}
```

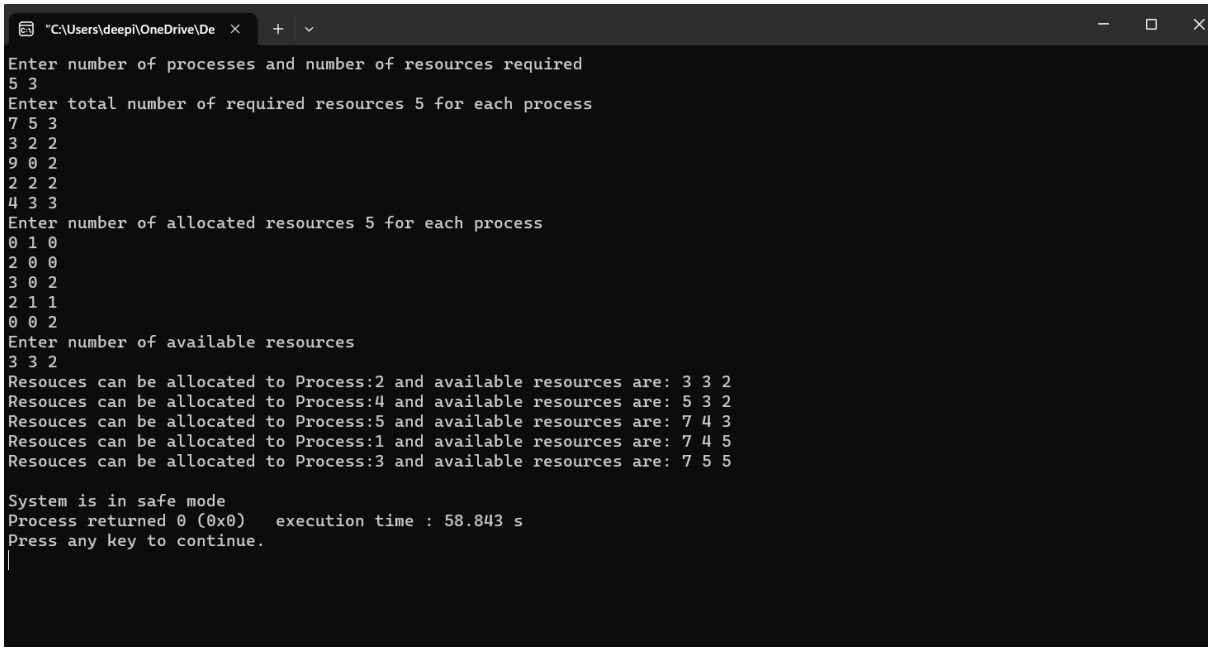
Output

```
Enter number of processes and resources required 5 3
Enter total number of resources required
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
Enter number of allocated resources
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
```

```
Enter number of alloted resources
3 3 2

Resources allocated to 2, available : 3 3 2
Resources allocated to 4, available : 5 3 2
Resources allocated to 5, available : 7 4 3
Resources allocated to 1, available : 7 4 5
Resources allocated to 3, available : 7 5 5

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

System is in safe mode
Sequence : 2 4 5 1 3
```

H/W

OUTPUT :


"C:\Users\deepi\OneDrive\De
Enter number of processes and number of resources required
5 3
Enter total number of required resources 5 for each process
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
Enter number of allocated resources 5 for each process
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
Enter number of available resources
3 3 2
Resouces can be allocated to Process:2 and available resources are: 3 3 2
Resouces can be allocated to Process:4 and available resources are: 5 3 2
Resouces can be allocated to Process:5 and available resources are: 7 4 3
Resouces can be allocated to Process:1 and available resources are: 7 4 5
Resouces can be allocated to Process:3 and available resources are: 7 5 5
System is in safe mode
Process returned 0 (0x0) execution time : 58.843 s
Press any key to continue.