NAME – AYUSH SHARMA
REG. NO.- 18BCE0172

ASSIGNMENT- 4

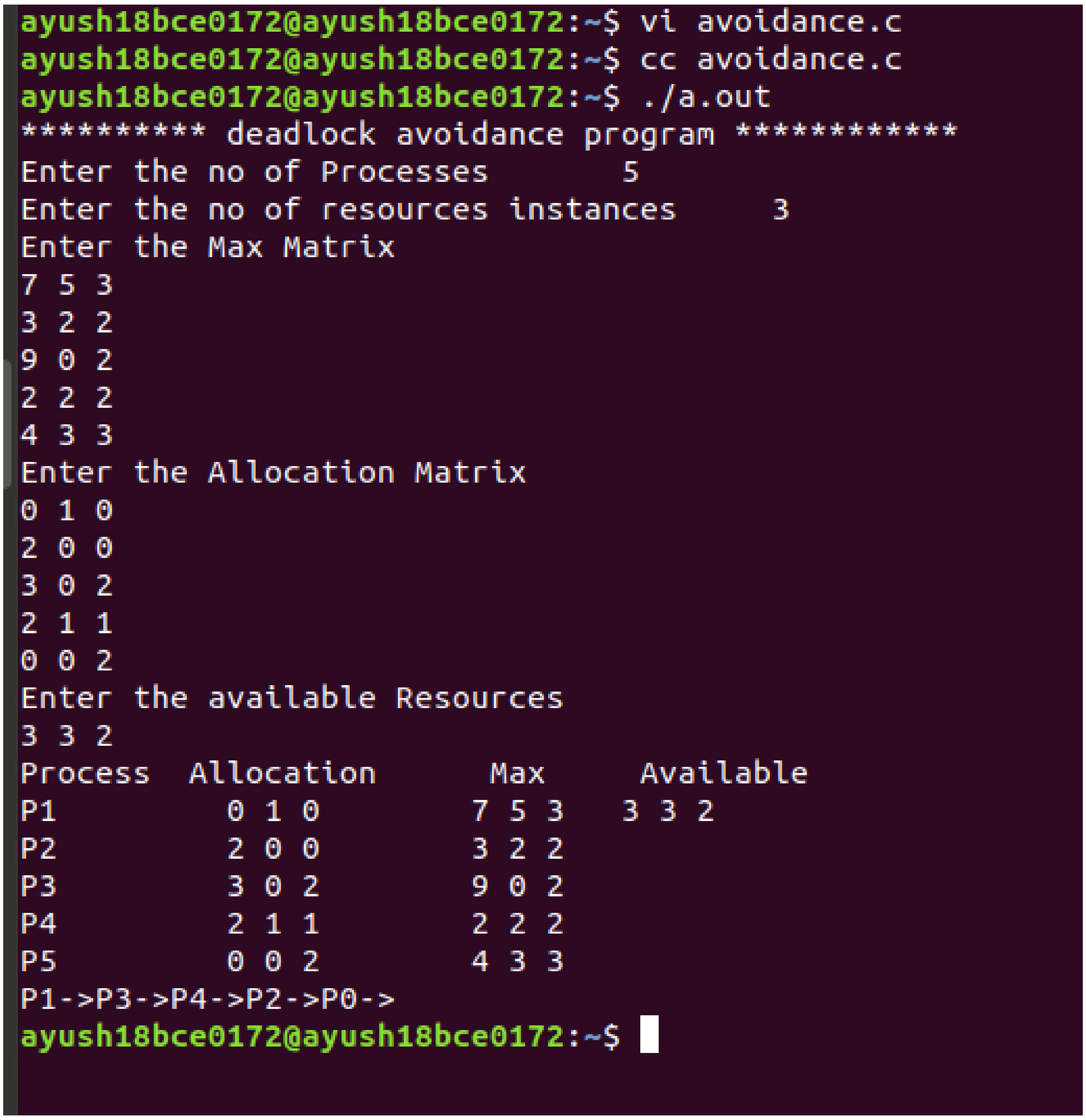
- **C PROGRAM FOR DEADLOCK AVOIDANCE**

CODE:

```
#include<stdio.h>
int max[100][100];
int alloc[100][100];
int need[100][100];
int avail[100];
int n,r;
void input();
void show();
void cal();
int main()
{
int i,j;
printf("********** deadlock avoidance program ************\n");
input();
show();
cal();
return 0;
}
void input()
{
int i,j;
printf("Enter the no of Processes\t");
scanf("%d",&n);
printf("Enter the no of resources instances\t");
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",&alloc[i][j]);
}
}
printf("Enter the available Resources\n");
for(j=0;j<r;j++)
{
scanf("%d",&avail[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 ",alloc[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 ",avail[j]);
}
}
}
void cal()
{
int finish[100],temp,need[100][100],flag=1,k,c1=0;
int safe[100];
int i,j;
for(i=0;i<n;i++)
{
finish[i]=0;
}
//find need matrix
for(i=0;i<n;i++)
{
for(j=0;j<r;j++)
{
need[i][j]=max[i][j]-alloc[i][j];
}
}
printf("\n");
```

```
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]<=avail[j]))
{
    c++;
if(c==r)
{
for(k=0;k<r;k++)
{
avail[k]+=alloc[i][j];
finish[i]=1;
flag=1;
      }
printf("P%d->",i);
if(finish[i]==1)
{
i=n;
}
      }
}
}
  }
  }
  for(i=0;i<n;i++)
  {
if(finish[i]==1)
{
c1++;
}
else
{
printf("P%d->",i);
}
}
if(c1==n)
{
printf("\n The system is in safe state");
}
else
{
printf("\n Process are in dead lock");
printf("\n System is in unsafe state");
}
}
```

OUTPUT:



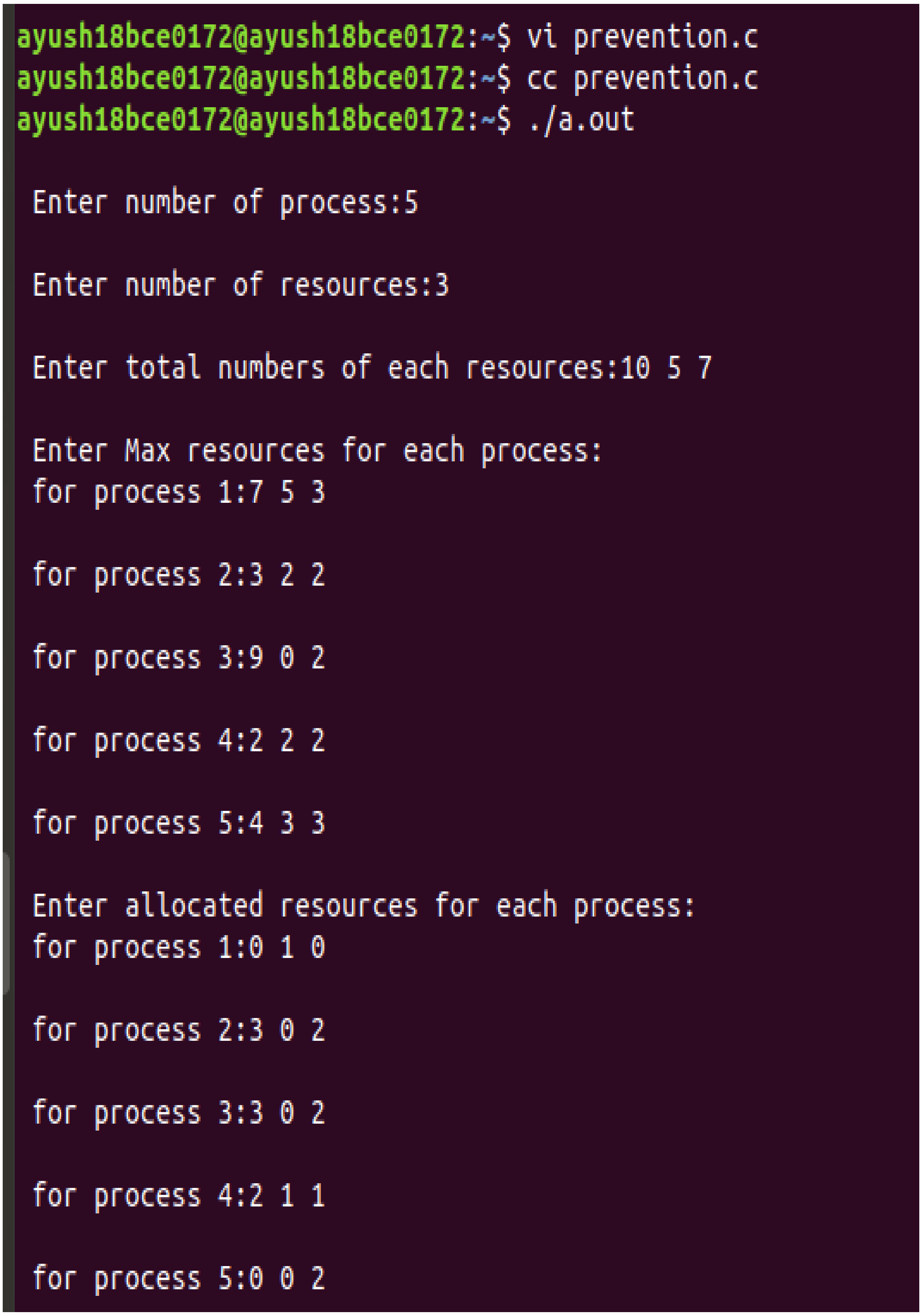
```
ayush18bce0172@ayush18bce0172:~$ vi avoidance.c
ayush18bce0172@ayush18bce0172:~$ cc avoidance.c
ayush18bce0172@ayush18bce0172:~$ ./a.out
********** deadlock avoidance program ************
Enter the no of Processes       5
Enter the no of resources instances     3
Enter the Max Matrix
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
Enter the Allocation Matrix
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
Enter the available Resources
3 3 2
Process  Allocation      Max      Available
P1          0 1 0       7 5 3    3 3 2
P2          2 0 0       3 2 2
P3          3 0 2       9 0 2
P4          2 1 1       2 2 2
P5          0 0 2       4 3 3
P1->P3->P4->P2->P0->
ayush18bce0172@ayush18bce0172:~$
```

- **C PROGRAM FOR DEADLOCK PREVENTION**

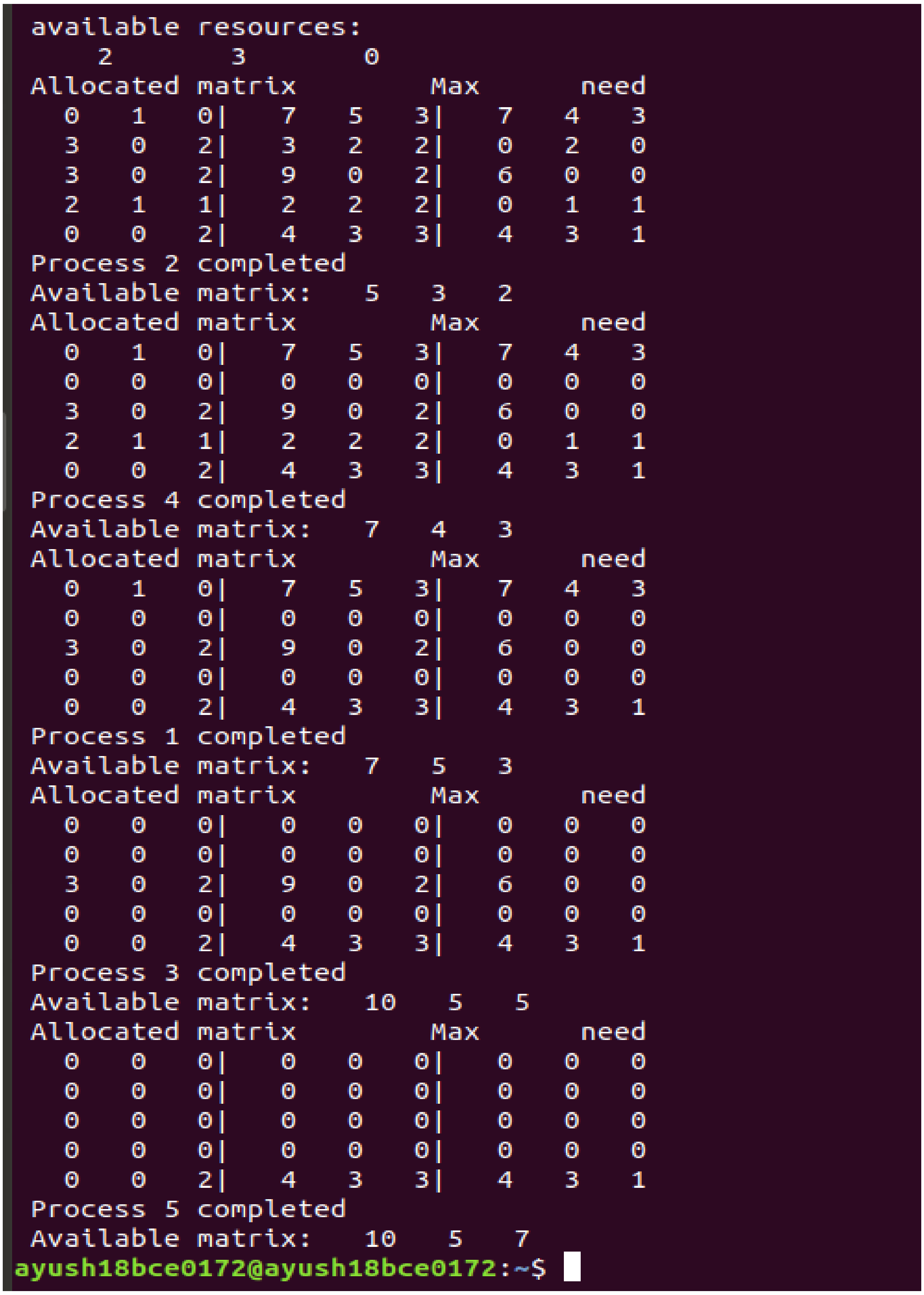
CODE:

```
#include< stdio.h>
void main()
{
 int allocated[15][15],max[15][15],need[15][15],avail[15],tres[15],work[15],flag[15];
 int pno,rno,i,j,prc,count,t,total;
 count=0;

 printf("\n Enter number of process:");
 scanf("%d",&pno);
 printf("\n Enter number of resources:");
 scanf("%d",&rno);
 for(i=1;i<=pno;i++)
 {
  flag[i]=0;
 }
 printf("\n Enter total numbers of each resources:");
 for(i=1;i<= rno;i++)
  scanf("%d",&tres[i]);


 printf("\n Enter Max resources for each process:");
 for(i=1;i<= pno;i++)
 {
  printf("\n for process %d:",i);
  for(j=1;j<= rno;j++)
   scanf("%d",&max[i][j]);
 }


 printf("\n Enter allocated resources for each process:");
 for(i=1;i<= pno;i++)
 {
  printf("\n for process %d:",i);
  for(j=1;j<= rno;j++)
   scanf("%d",&allocated[i][j]);


 }

 printf("\n available resources:\n");
 for(j=1;j<= rno;j++)
 {
  avail[j]=0;
  total=0;
  for(i=1;i<= pno;i++)
  {
   total+=allocated[i][j];
  }
  avail[j]=tres[j]-total;
  work[j]=avail[j];
  printf("     %d \t",work[j]);
 }
 do
 {
 for(i=1;i<= pno;i++)
 {
  for(j=1;j<= rno;j++)
  {
   need[i][j]=max[i][j]-allocated[i][j];
  }
 }

 printf("\n Allocated matrix         Max       need");
 for(i=1;i<= pno;i++)
 {
  printf("\n");
  for(j=1;j<= rno;j++)
  {
   printf("%4d",allocated[i][j]);
  }
  printf("|");
  for(j=1;j<= rno;j++)
  {
   printf("%4d",max[i][j]);
  }
   printf("|");
  for(j=1;j<= rno;j++)
  {
   printf("%4d",need[i][j]);
  }
 }
  prc=0;

  for(i=1;i<= pno;i++)
  {
   if(flag[i]==0)
   {
    prc=i;

    for(j=1;j<= rno;j++)
    {
     if(work[j]< need[i][j])
     {
       prc=0;
       break;
     }
    }
   }

   if(prc!=0)
   break;
  }

  if(prc!=0)
  {
   printf("\n Process %d completed",i);
   count++;
   printf("\n Available matrix:");
   for(j=1;j<= rno;j++)
   {
    work[j]+=allocated[prc][j];
    allocated[prc][j]=0;
    max[prc][j]=0;
    flag[prc]=1;
    printf("   %d",work[j]);
   }
  }

 }while(count!=pno&&prc!=0);

 if(count==pno)
  printf("\nThe system is in a safe state!!");
 else
  printf("\nThe system is in an unsafe state!!");

 }
```

OUTPUT:



```
ayush18bce0172@ayush18bce0172:~$ vi prevention.c
ayush18bce0172@ayush18bce0172:~$ cc prevention.c
ayush18bce0172@ayush18bce0172:~$ ./a.out

 Enter number of process:5

 Enter number of resources:3

 Enter total numbers of each resources:10 5 7

 Enter Max resources for each process:
 for process 1:7 5 3

 for process 2:3 2 2

 for process 3:9 0 2

 for process 4:2 2 2

 for process 5:4 3 3

 Enter allocated resources for each process:
 for process 1:0 1 0

 for process 2:3 0 2

 for process 3:3 0 2

 for process 4:2 1 1

 for process 5:0 0 2
```



```
available resources:
    2       3       0
Allocated matrix        Max       need
   0    1    0|    7    5    3|    7    4    3
   3    0    2|    3    2    2|    0    2    0
   3    0    2|    9    0    2|    6    0    0
   2    1    1|    2    2    2|    0    1    1
   0    0    2|    4    3    3|    4    3    1
Process 2 completed
Available matrix:    5    3    2
Allocated matrix        Max       need
   0    1    0|    7    5    3|    7    4    3
   0    0    0|    0    0    0|    0    0    0
   3    0    2|    9    0    2|    6    0    0
   2    1    1|    2    2    2|    0    1    1
   0    0    2|    4    3    3|    4    3    1
Process 4 completed
Available matrix:    7    4    3
Allocated matrix        Max       need
   0    1    0|    7    5    3|    7    4    3
   0    0    0|    0    0    0|    0    0    0
   3    0    2|    9    0    2|    6    0    0
   0    0    0|    0    0    0|    0    0    0
   0    0    2|    4    3    3|    4    3    1
Process 1 completed
Available matrix:    7    5    3
Allocated matrix        Max       need
   0    0    0|    0    0    0|    0    0    0
   0    0    0|    0    0    0|    0    0    0
   3    0    2|    9    0    2|    6    0    0
   0    0    0|    0    0    0|    0    0    0
   0    0    2|    4    3    3|    4    3    1
Process 3 completed
Available matrix:    10    5    5
Allocated matrix        Max       need
   0    0    0|    0    0    0|    0    0    0
   0    0    0|    0    0    0|    0    0    0
   0    0    0|    0    0    0|    0    0    0
   0    0    0|    0    0    0|    0    0    0
   0    0    2|    4    3    3|    4    3    1
Process 5 completed
Available matrix:    10    5    7
ayush18bce0172@ayush18bce0172:~$
```

- **C PROGRAM FOR DEADLOCK DETECTION**

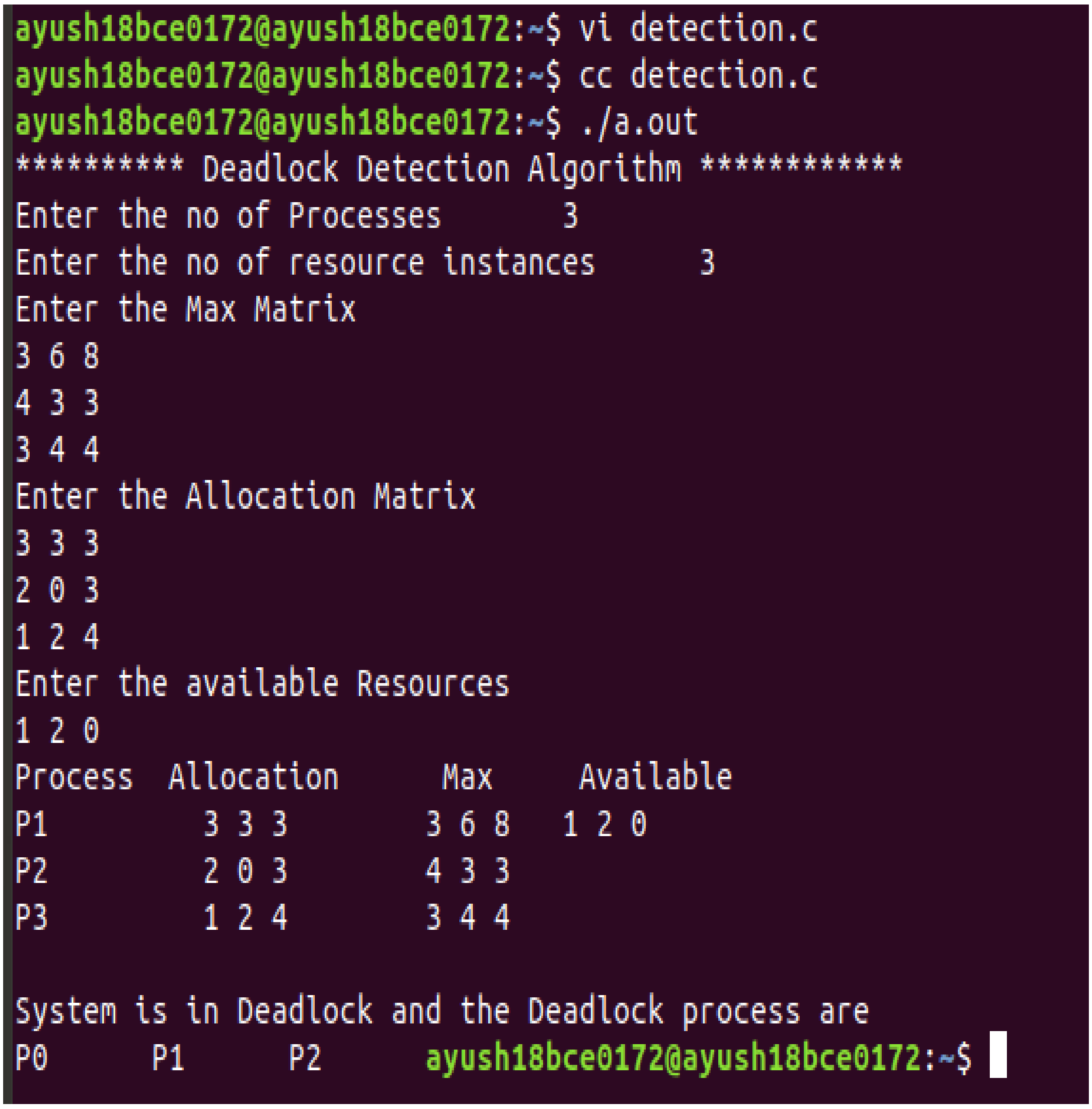
CODE:

```
#include<stdio.h>
int max[100][100];
int alloc[100][100];
int need[100][100];
int avail[100];
int n,r;
void input();
void show();
void cal();
int main()
{
	int i,j;
	printf("********** Deadlock Detection Algorithm ************\n");
	input();
	show();
	cal();
	return 0;
}
void input()
{
	int i,j;
	printf("Enter the no of Processes\t");
	scanf("%d",&n);
	printf("Enter the no of resource instances\t");
	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",&alloc[i][j]);
		}
	}
	printf("Enter the available Resources\n");
	for(j=0;j<r;j++)
	{
		scanf("%d",&avail[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 ",alloc[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 ",avail[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;
	}
	//find need matrix
	for(i=0;i<n;i++)
	{
		for(j=0;j<r;j++)
		{
			need[i][j]=max[i][j]-alloc[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]<=avail[j]))
                {
                    c++;
                    if(c==r)
                    {
                        for(k=0;k<r;k++)
                        {
                            avail[k]+=alloc[i][j];
                            finish[i]=1;
                            flag=1;
                        }
                        //printf("\nP%d",i);
                        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:



```
ayush18bce0172@ayush18bce0172:~$ vi detection.c
ayush18bce0172@ayush18bce0172:~$ cc detection.c
ayush18bce0172@ayush18bce0172:~$ ./a.out
********** Deadlock Detection Algorithm ************
Enter the no of Processes       3
Enter the no of resource instances      3
Enter the Max Matrix
3 6 8
4 3 3
3 4 4
Enter the Allocation Matrix
3 3 3
2 0 3
1 2 4
Enter the available Resources
1 2 0
Process  Allocation      Max     Available
P1          3 3 3       3 6 8   1 2 0
P2          2 0 3       4 3 3
P3          1 2 4       3 4 4

System is in Deadlock and the Deadlock process are
P0      P1      P2      ayush18bce0172@ayush18bce0172:~$ 
```