Course Name: Operating systems

LAB: 08

Talal Khan

Roll: DT-22043

PROGRAM:

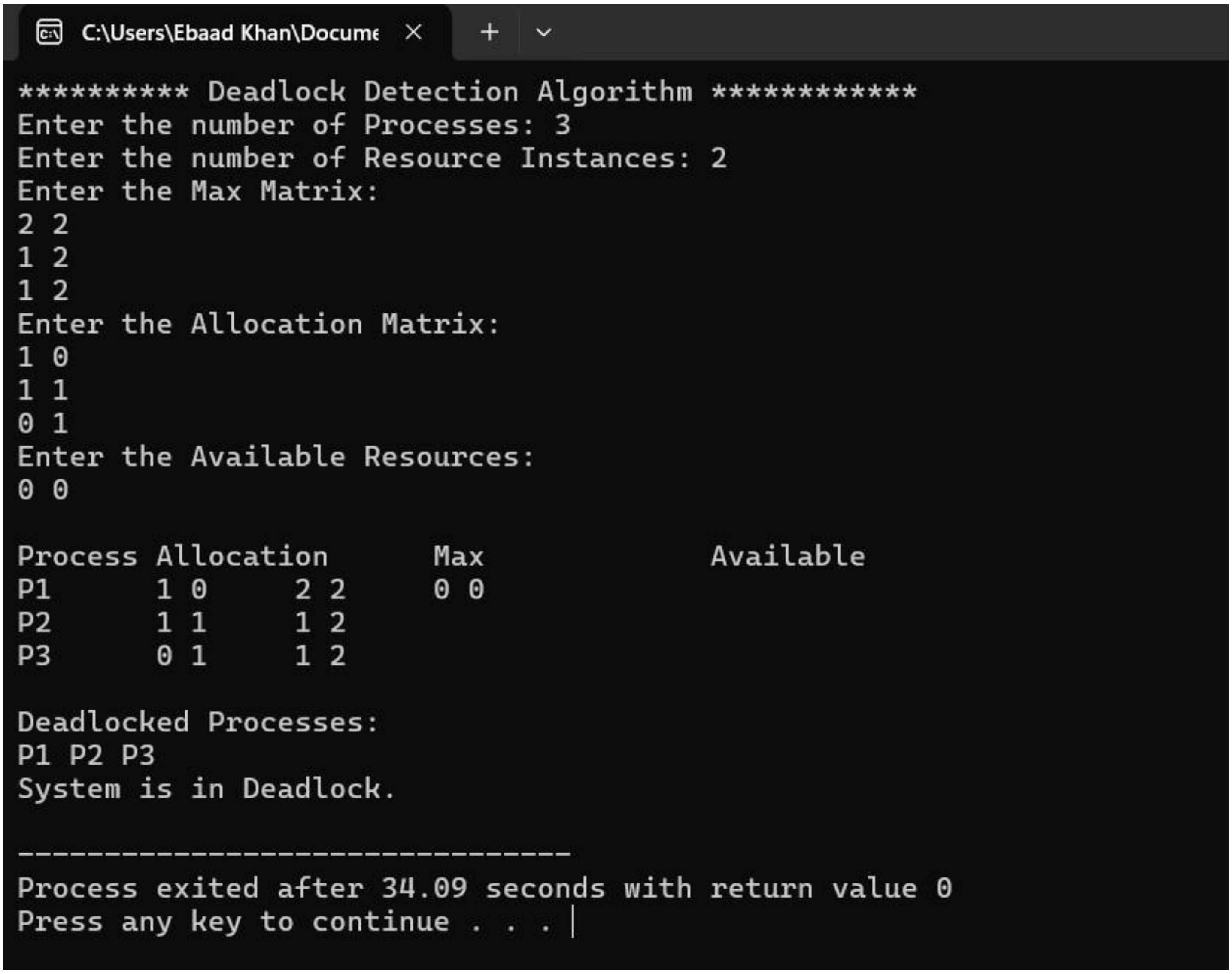
```
#include <stdio.h>


int    max[100][100],    alloc[100][100],    need[100][100];    int

avail[100];

int n, r;


void    input();

void    show();

void cal();


int    main()    {    printf("**********    Deadlock    Detection    Algorithm

   ************\n"); input(); show();

   cal(); return 0;

}
void input() { int i, j; printf("Enter the number of Processes: ");

   scanf("%d",  &n);  printf("Enter  the  number  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", &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("\nProcess\tAllocation\tMax\t\tAvailable\n");

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

      printf("P%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]);

            }

        }

        printf("\n");

    }

 }

void cal() { int flnish[100],

 dead[100]; int i, j, k, flag = 1, c1 =

 0;

    // Calculate need matrix for(i = 0; i < n;

    i++) { flnish[i] = 0; 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

            canExecute = 1; if(flnish[i] == 0)

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

```
        if(need[i][j]     >     avail[j])     {

        canExecute = 0; break;

                }

            }


                if(canExecute) {

                for(j = 0; j < r; j++) { avail[j] +=

                    alloc[i][j];

                } flnish[i] = 1;

                flag = 1;


            }

        }

    }

}


int   deadlock   =   0;   printf("\nDeadlocked

Processes:\n");   for(i   =   0;   i   <   n;   i++)   {

if(flnish[i] == 0) { printf("P%d ", i + 1);

        deadlock = 1;

    }

}


if(deadlock  ==  0)  {  printf("No  Deadlock  Detected.  System  is  in  Safe

    State.\n");
```

```
   } else { printf("\nSystem is in Deadlock.\n");

   }

}
```

OUTPUT:



```
********** Deadlock Detection Algorithm ************
Enter the number of Processes: 3
Enter the number of Resource Instances: 2
Enter the Max Matrix:
2 2
1 2
1 2
Enter the Allocation Matrix:
1 0
1 1
0 1
Enter the Available Resources:
0 0

Process Allocation      Max             Available
P1      1 0     2 2     0 0
P2      1 1     1 2
P3      0 1     1 2

Deadlocked Processes:
P1 P2 P3
System is in Deadlock.

--------------------------------
Process exited after 34.09 seconds with return value 0
Press any key to continue . . .
```