ITE2002-OPERATING SYSTEM

LAB

WINTER SEM 20-21

Assessment – 4

CAT-2

Name :Pravin G

Reg No :19BIT0393

Slot :L41-42

Algorithms:

```
Step 1:-
Read no of procees(n),no of resources(m);
Read no of Maximum resources
Do flag[i] for i=0 to n
Step2:-
Find processs pi such that flag[i]=0 and needi<=Available
Step 3:
If exist
    Flag[i]=1
    Available=available+allocate
    Go to step2
Else
    Go to step 4
Step4
Flag[i]=1 for all I then safe state otherwise unsafe state
Display Sequenve

Step 5
if request <need goto step 6 else don't grant
Step 6
Request<avail goto step 7

Step 7
Avail-=request
Allocation+=request
Need-=request
Then check bankers
If safe grant else don't grant
```

Code:

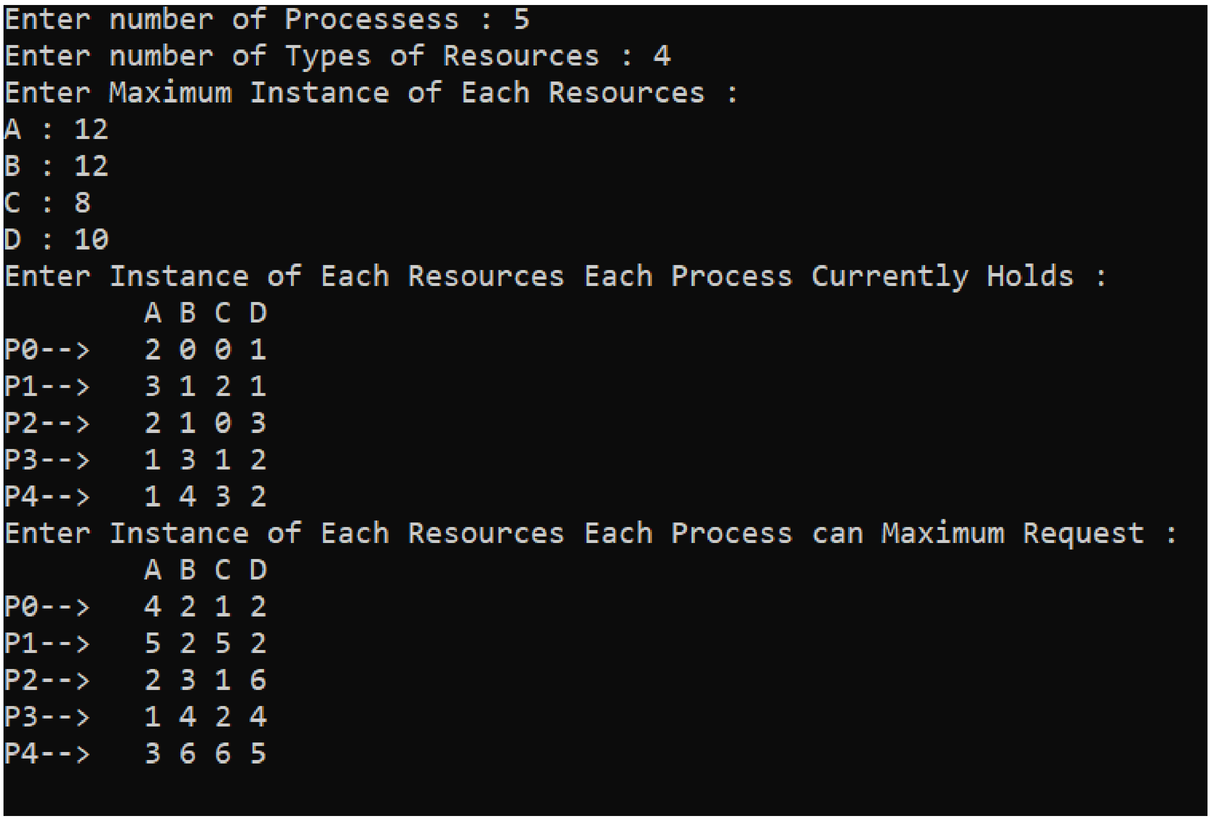
```c
#include <stdio.h>
int i, j;
int n;
int m;

int instance[10];

int max[10][10];
int allocation[10][10];

int available[10];
int availback[10];
int request[10];

int need[10][10];

int sequence[10];
int availseq[10][10];
int flag1[10];
int doneprocess;

void read()
{
  printf("Enter number of Processess : ");
  scanf("%d", &n);

  printf("Enter number of Types of Resources : ");
  scanf("%d", &m);

  printf("Enter Maximum Instance of Each Resources : \n");
  for (i = 0; i < m; i++)
  {
    printf("%c : ", i + 'A');
    scanf("%d", &instance[i]);
  }
  printf("Enter Instance of Each Resources Each Process Curr
ently Holds :\n\t");
  for (i = 0; i < m; i++)
    printf("%c ", i + 'A');
  printf("\n");
  for (i = 0; i < n; i++)
  {
    printf("P%d-->   ", i);
    for (j = 0; j < m; j++)
      scanf("%d", &allocation[i][j]);
  }
  printf("Enter Instance of Each Resources Each Process can 
Maximum Request :\n\t");
  for (i = 0; i < m; i++)
    printf("%c ", i + 'A');
  printf("\n");
  for (i = 0; i < n; i++)
  {
    printf("P%d-->   ", i);
    for (j = 0; j < m; j++)
      scanf("%d", &max[i][j]);
  }
}

void dispavail()
{
  printf("\nAvailable Resources : ");
  for (i = 0; i < m; i++)
  {
    available[i] = instance[i];
    for (j = 0; j < n; j++)
      available[i] -= allocation[j][i];
    availback[i] = available[i];
    printf("%d ", available[i]);
  }
}
void dispneed()
```

```
{
  printf("\n\nNeed of Each Resources of Each Process :\n\t")
;
  for (i = 0; i < m; i++)
    printf("%c ", i + 'A');

  for (i = 0; i < n; i++)
  {
    printf("\nP%d\t", i);
    for (j = 0; j < m; j++)
    {
      need[i][j] = max[i][j] - allocation[i][j];
      printf("%d ", need[i][j]);
    }
  }
}
int check()
{
  int count = 0;
  for (i = 0; i < n; i++)
  {
    if (flag1[i] == 1)
      count++;
  }
  if (count == n)
    return 0;
  else if (doneprocess == count)
    return -1;
  else
  {
    doneprocess = count;
    return 1;
  }
}
int bankers()
{
  int flag2;
  int ans = 0;
  int idx = 0;
  doneprocess = 0;

  for (i = 0; i < n; i++)
    flag1[i] = 0;

  do
  {
    for (i = 0; i < n; i++)
    {
      if (flag1[i] == 0)
      {
        flag2 = 0;
        for (j = 0; j < m; j++)
        {
          if (need[i][j] > available[j])
          {
            flag2 = 1;
            break;
          }
        }
        if (flag2 == 0)
        {
          sequence[idx] = i;
          flag1[i] = 1;
          for (j = 0; j < m; j++)
          {
            availseq[idx][j] = available[j];
            available[j] += allocation[i][j];
          }
          idx++;
        }
      }
    }
  } while ((ans = check()) == 1);

  return ans;
}
```

```c
void checkrequest()
{
  int id, ans;
  printf("\nEnter Id of Process to Request Resource :-");
  scanf("%d", &id);
  printf("\nEnter Requesdted Resources For Each Type :\n");
  for (i = 0; i < m; i++)
  {
    available[i] = availback[i];
    printf("%c : ", i + 'A');
    scanf("%d", &request[i]);
  }
  for (i = 0; i < m; i++)
  {
    if (request[i] + allocation[id][i] > max[id][i])
    {
      printf("\nRequest Can not be Granted \nRequest is More
 than Maximum Request Count");
      return;
    }
    allocation[id][i] += request[i];
    available[i] -= request[i];
    need[id][i] -= request[i];
  }
  ans = bankers();
  if (ans == -1)
  {
    printf("\nRequest Can not be granted\nUnsafe State will
Occur\n");
    printf("\nUnSafe Sequence : <");
    for (i = 0; i < n; i++)
    {
      printf(" P%d(", sequence[i]);
      for (j = 0; j < m; j++)
        printf(" %d", availseq[i][j]);
      printf(" )");
    }
    printf(">\nAfter this we can not execute any process\n")
;
  }
  else
  {
    printf("\nRequest Can be Granted\n");
    printf("\nSafe Sequence : <");
    for (i = 0; i < n; i++)
    {
      printf(" P%d(", sequence[i]);
      for (j = 0; j < m; j++)
        printf(" %d", availseq[i][j]);
      printf(" )");
    }

    printf(" >");
  }
}
int main()
{
  int ans;
  read();

  dispneed();
  dispavail();

  ans = bankers();
  if (ans == -1)
  {
    printf("\nSystem in unsafe state\nDeadlock may occur");
    printf("\nUnSafe Sequence : <");
    for (i = 0; i < n; i++)
    {
      printf(" P%d(", sequence[i]);
      for (j = 0; j < m; j++)
        printf(" %d", availseq[i][j]);
      printf(" )");
    }
```

```c
    printf(">\nAfter this we can not execute any process");
  }
  else
  {
    printf("\nSystem in Safe state\nDeadlock will not occur\
n");
    printf("\nSafe Sequence : <");
    for (i = 0; i < n; i++)
    {
      printf(" P%d(", sequence[i]);
      for (j = 0; j < m; j++)
        printf(" %d", availseq[i][j]);
      printf(" )");
    }

    printf(" >");
  }
  checkrequest();

  return 0;
}
```

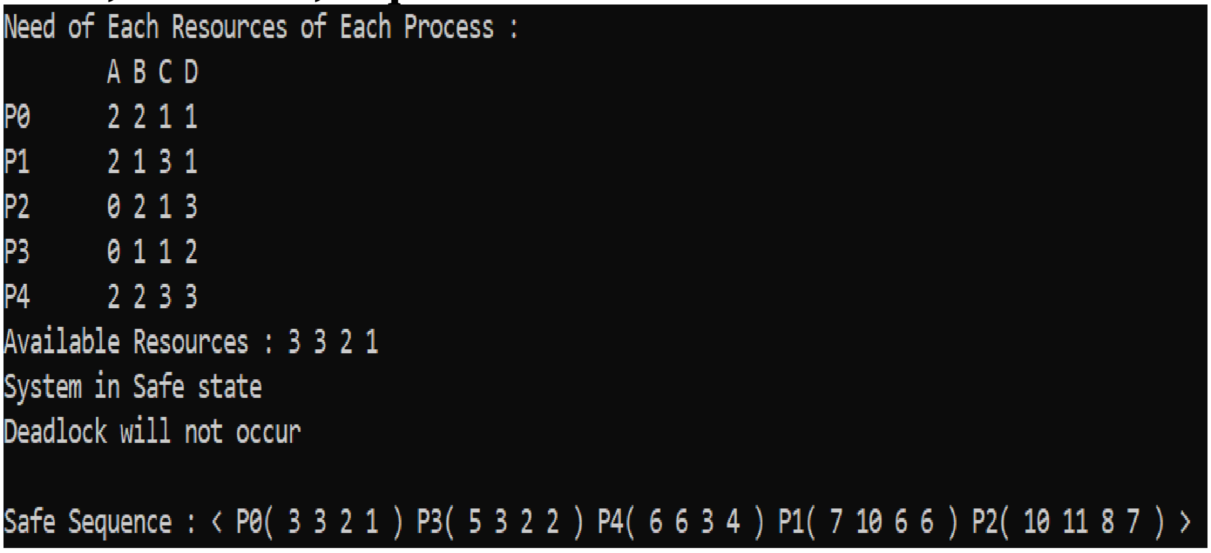
Input 1



```
Enter number of Processess : 5
Enter number of Types of Resources : 4
Enter Maximum Instance of Each Resources :
A : 12
B : 12
C : 8
D : 10
Enter Instance of Each Resources Each Process Currently Holds :
        A B C D
P0-->   2 0 0 1
P1-->   3 1 2 1
P2-->   2 1 0 3
P3-->   1 3 1 2
P4-->   1 4 3 2
Enter Instance of Each Resources Each Process can Maximum Request :
        A B C D
P0-->   4 2 1 2
P1-->   5 2 5 2
P2-->   2 3 1 6
P3-->   1 4 2 4
P4-->   3 6 6 5
```

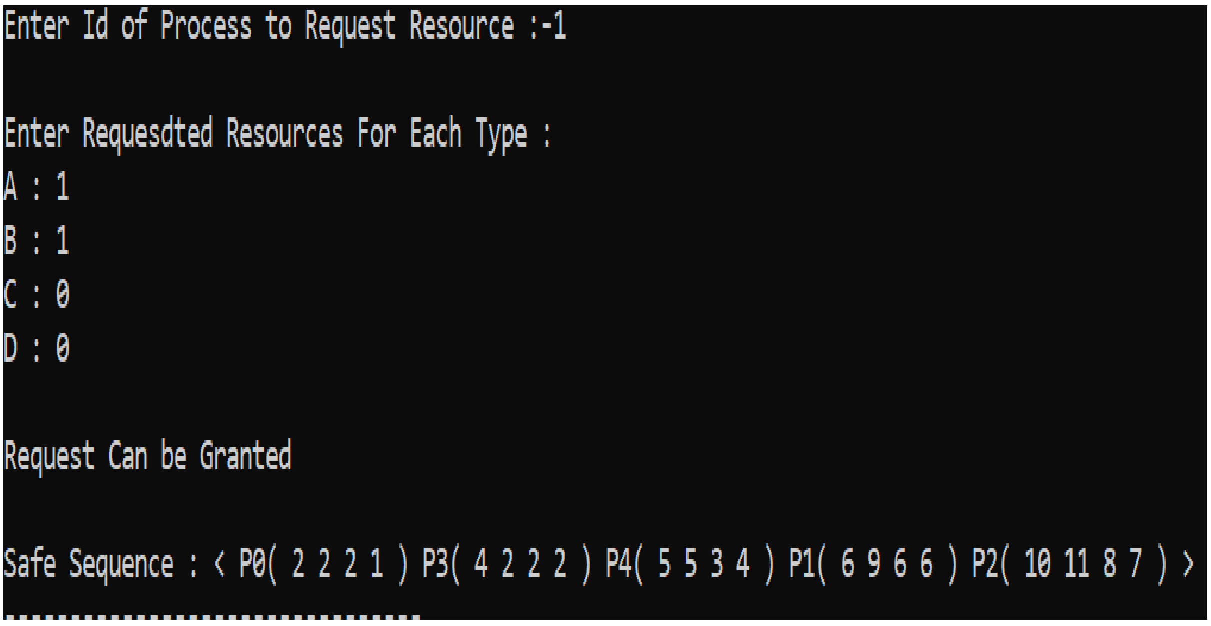
Output

Need,Available,Sequence:-



```
Need of Each Resources of Each Process :
        A B C D
P0      2 2 1 1
P1      2 1 3 1
P2      0 2 1 3
P3      0 1 1 2
P4      2 2 3 3
Available Resources : 3 3 2 1
System in Safe state
Deadlock will not occur

Safe Sequence : < P0( 3 3 2 1 ) P3( 5 3 2 2 ) P4( 6 6 3 4 ) P1( 7 10 6 6 ) P2( 10 11 8 7 ) >
```

Request :-



```
Enter Id of Process to Request Resource :-1

Enter Requesdted Resources For Each Type :
A : 1
B : 1
C : 0
D : 0

Request Can be Granted

Safe Sequence : < P0( 2 2 2 1 ) P3( 4 2 2 2 ) P4( 5 5 3 4 ) P1( 6 9 6 6 ) P2( 10 11 8 7 ) >
```