Dr. D.Y.Patil Arts, Commerce , Science College,

Pimpri, Pune-18

T.Y.B.Sc(Comp. Sci) 2022-23

Operating System-II

Solution of Practical Assignment 1: Banker's Algorithm

Slot 1

Add the following functionalities in your program

a) Accept Available

b) Display Allocation, Max

c) Display the contents of need matrix

d) Display Available

```
#include<stdio.h>
int nop,nor,A[10][10],M[10][10],Av[10],N[10][10],finish[10];
void acceptdata(int x[10][10])
{
   int i,j;
   for(i=0;i<nop;i++)
     {
     printf("P%d\n",i);
       for(j=0;j<nor;j++)
      {
        printf("%c: ",65+j);
           scanf("%d",&x[i][j]);
      }
     }
}
void acceptav()
{
  int i;
  for(i=0;i<nor;i++)
  {
    printf("%c: ",65+i);
    scanf("%d",&Av[i]);
  }
}
void calcneed()
{
  int i,j;
  for(i=0;i<nop;i++)
    for(j=0;j<nor;j++)
      N[i][j]=M[i][j]-A[i][j];

}
void displaydata()
{
      int i,j;
      printf("\n\tAllocation \t\tMax\t\tNeed\n\t");
      for(i=0;i<3;i++)
      {
           for(j=0;j<nor;j++)
                 printf("%4c",65+j);
           printf("\t");
     }
     for(i=0;i<nop;i++)
     {
           printf("\nP%d\t",i);
           for(j=0;j<nor;j++)
                 printf("%4d",A[i][j]);
           printf("\t");
           for(j=0;j<nor;j++)
                 printf("%4d",M[i][j]);
           printf("\t");
           for(j=0;j<nor;j++)
                 printf("%4d",N[i][j]);
     }
     printf("\navailable");
     for(i=0;i<nor;i++)
           printf("%4d",Av[i]);
}
int checkneed(int pno)
{
     int i;
     for(i=0;i<nor;i++)
           if(N[pno][i]>Av[i])
                 return 0;
     return 1;
}

main()
{
   printf("\nEnter No of Processes: ");
   scanf("%d",&nop);
   printf("\nEnter No. of Resources: ");
   scanf("%d",&nor);
   printf("\nEnter Allocation Matrix: ");
   acceptdata(A);
  printf("\nEnter Max Matrix: ");
   acceptdata(M);
  printf("\nEnter Availability:");
  acceptav();
  calcneed();
  displaydata();
}
```

Add the following functionalities in your program
a) Accept Available
b) Display Allocation, Max
c) Display the contents of need matrix
d) Display Available
Implement Bankers Algorithm

```
#include<stdio.h>
int nop,nor,A[10][10],M[10][10],Av[10],N[10][10],finish[10];
void acceptdata(int x[10][10])
{
   int i,j;
   for(i=0;i<nop;i++)
     {
     printf("P%d\n",i);
       for(j=0;j<nor;j++)
     {
       printf("%c: ",65+j);
          scanf("%d",&x[i][j]);
     }
     }
}
void acceptav()
{
  int i;
  for(i=0;i<nor;i++)
  {
    printf("%c: ",65+i);
    scanf("%d",&Av[i]);
  }
}
void calcneed()
{
  int i,j;
  for(i=0;i<nop;i++)
    for(j=0;j<nor;j++)
      N[i][j]=M[i][j]-A[i][j];

}
void displaydata()
{
      int i,j;
      printf("\n\tAllocation \t\tMax\t\tNeed\n\t");
      for(i=0;i<3;i++)
      {
           for(j=0;j<nor;j++)
                 printf("%4c",65+j);
           printf("\t");
      }
      for(i=0;i<nop;i++)
      {
           printf("\nP%d\t",i);
```

```
                  for(j=0;j<nor;j++)
                        printf("%4d",A[i][j]);
                  printf("\t");
                  for(j=0;j<nor;j++)
                        printf("%4d",M[i][j]);
                  printf("\t");
                  for(j=0;j<nor;j++)
                        printf("%4d",N[i][j]);
            }
            printf("\navailable");
            for(i=0;i<nor;i++)
                  printf("%4d",Av[i]);
}
int checkneed(int pno)
{
            int i;
            for(i=0;i<nor;i++)
                  if(N[pno][i]>Av[i])
                        return 0;
            return 1;
}

void banker()
{
            int p=0,j=0,k=0,flag=0,safe[10];
            while(flag<2)
            {
                  if(!finish[p])
                  {
                        printf("\n\nNeed of process P%d (,",p);
                        for(j=0;j<nor;j++)
                              printf("%d,",N[p][j]);
                        if(checkneed(p))
                        {
                              printf(") <= available (");
                              for(j=0;j<nor;j++)
                                    printf("%d,",Av[j]);
                              printf(")");

                              printf("\nNeed is Satsified, So process P%d can be
granted requiered resources.\n After P%d finishes, it will realease all
the resources.",p,p);
                              for(j=0;j<nor;j++)
                                    Av[j]=Av[j]+A[p][j];

                              printf("New Availble=");
                              for(j=0;j<nor;j++)
                                    printf("%d ",Av[j]);
                              finish[p]=1;
                              safe[k++]=p;

                        }
                        else
                        {
                              printf(") >  available (");
                              for(j=0;j<nor;j++)
```

```
                            printf("%d,",Av[j]);
                      printf(")");

                      printf("\nNeed is not Satsified, So process P%d
cannot be granted required resources.\n process P%d has to wait.",p,p);
                 }
           }
           if((p+1)%nop==0)
                 flag++;
           p=(p+1)%nop;
      }//while
       if(k==nop)
      {
          printf("\nSystem is in safe state...");
          printf("\nSafe Sequence: ");
          for(j=0;j<k;j++)
           printf("P%d->",safe[j]);
       }
      else
     printf("\nSystem is not in safe state....");
}
main()
{
   printf("\nEnter No of Processes: ");
   scanf("%d",&nop);
   printf("\nEnter No. of Resources: ");
   scanf("%d",&nor);
   printf("\nEnter Allocation Matrix: ");
   acceptdata(A);
  printf("\nEnter Max Matrix: ");
   acceptdata(M);
  printf("\nEnter Availability:");
  acceptav();
  calcneed();
  displaydata();
  banker();
}
```

Slot 2

Modify above program so as to include the following:

a) Accept Request for a process

b) Resource request algorithm

c) Safety algorithm Consider a system with ‘n’ processes and ‘m’ resource types.

Accept number of instances for every resource type. For each process accept the allocation and maximum requirement matrices. Write a program to display the contents of need matrix and to check if the given request of a process can be granted immediately or not.

```
#include<stdio.h>
int
nop,nor,Rprocess,A[10][10],M[10][10],Av[10],N[10][10],R[10],finish[10];
void acceptdata(int x[10][10])
{
      int i,j;
      for(i=0;i<nop;i++)
      {
            printf("P%d\n",i);
            for(j=0;j<nor;j++)
            {
                  printf("%c: ",65+j);
                  scanf("%d",&x[i][j]);
            }
      }
}
void acceptav()
{
      int i;
      for(i=0;i<nor;i++)
      {
            printf("%c: ",65+i);
            scanf("%d",&Av[i]);
      }
}
void acceptrequest()
{
      int i;
      printf("\nEnter the Process for which request has arrived :P");
      scanf("%d",&Rprocess);
      printf("\nEnter the request for process: ");
      for(i=0;i<nor;i++)
      {
            printf("%c: ",65+i);
            scanf("%d",&R[i]);
      }


}
```

```
void calcneed()
{
      int i,j;
      for(i=0;i<nop;i++)
            for(j=0;j<nor;j++)
                  N[i][j]=M[i][j]-A[i][j];

}
void displaydata()
{
      int i,j;
      printf("\n\tAllocation \t\tMax\t\tNeed\n\t");
      for(i=0;i<3;i++)
      {
            for(j=0;j<nor;j++)
                  printf("%4c",65+j);
            printf("\t");
      }
      for(i=0;i<nop;i++)
      {
            printf("\nP%d\t",i);
            for(j=0;j<nor;j++)
                  printf("%4d",A[i][j]);
            printf("\t");
            for(j=0;j<nor;j++)
                  printf("%4d",M[i][j]);
            printf("\t");
            for(j=0;j<nor;j++)
                  printf("%4d",N[i][j]);
      }
      printf("\navailable");
      for(i=0;i<nor;i++)
            printf("%4d",Av[i]);
}
int checkneed(int pno)
{
      int i;
      for(i=0;i<nor;i++)
            if(N[pno][i]>Av[i])
                  return 0;
      return 1;
}
void resourcerequest()
{
      int i;

      for(i=0;i<nor;i++)
      {


            if(R[i]>N[Rprocess][i])
                  break;
      }
      if(i==nor)
      {
            for(i=0;i<nor;i++)
            {
                if(R[i]>Av[i])
                      break;
           }

      }
      if(i==nor)
      {
           printf("\nRequest<=Need \n Request<=Available \n Both
Condition are true");
           printf("\nThen system Pretends to fulfill request , then
modify resourse allocation state");
           for(i=0;i<nor;i++)
           {
                Av[i]=Av[i]-R[i];
                A[Rprocess][i]=A[Rprocess][i]+R[i];
                N[Rprocess][i]=N[Rprocess][i]-R[i];
           }
           displaydata();
      }
      else
      {
           printf("\nRequest<=Need \n Request<=Available \n Condition is
not true");
           printf("\nSo request cannot be satisfied!");
      }

}

main()
{
      printf("\nEnter No of Processes: ");
      scanf("%d",&nop);
      printf("\nEnter No. of Resources: ");
      scanf("%d",&nor);
      printf("\nEnter Allocation Matrix: ");
      acceptdata(A);
      printf("\nEnter Max Matrix: ");
      acceptdata(M);
      printf("\nEnter Availability:");
      acceptav();
      calcneed();
      displaydata();
      acceptrequest();
      resourcerequest();
      banker();
}
/*
[root@localhost ~]# cc resoucerequest.c
[root@localhost ~]# ./a.out

Enter No of Processes: 5

Enter No. of Resources: 3

Enter Allocation Matrix: P0
A: 0
```

```
B: 1
C: 0
P1
A: 2
B: 0
C: 0
P2
A: 3
B: 0
C: 2
P3
A: 2
B: 1
C: 1
P4
A: 0
B: 0
C: 2

Enter Max Matrix: P0
A: 7
B: 5
C: 3
P1
A: 3
B: 2
C: 2
P2
A: 9
B: 0
C: 2
P3
A: 2
B: 2
C: 2
P4
A: 4
B: 3
C: 3

Enter Availability:A: 3
B: 3
C: 2

        Allocation              Max             Need
           A   B   C       A   B   C       A   B   C
P0         0   1   0       7   5   3       7   4   3
P1         2   0   0       3   2   2       1   2   2
P2         3   0   2       9   0   2       6   0   0
P3         2   1   1       2   2   2       0   1   1
P4         0   0   2       4   3   3       4   3   1
available   3   3   2
Enter the Process for which request has arrived :P1

Enter the request for process: A: 1
B: 0
```

```
C: 2

Request<=Need
 Request<=Available
 Both Condition are true
Then system Pretends to fulfill request , then modify resourse
allocation state
        Allocation              Max             Need
           A   B   C       A   B   C       A   B   C
P0         0   1   0       7   5   3       7   4   3
P1         3   0   2       3   2   2       0   2   0
P2         3   0   2       9   0   2       6   0   0
P3         2   1   1       2   2   2       0   1   1
P4         0   0   2       4   3   3       4   3   1
available   2   3   0


*/
```

Resource Request and banker

```
#include<stdio.h>
int
nop,nor,Rprocess,A[10][10],M[10][10],Av[10],N[10][10],R[10],finish[10];
void acceptdata(int x[10][10])
{
      int i,j;
      for(i=0;i<nop;i++)
      {
            printf("P%d\n",i);
            for(j=0;j<nor;j++)
            {
                  printf("%c: ",65+j);
                  scanf("%d",&x[i][j]);
            }
      }
}
void acceptav()
{
      int i;
      for(i=0;i<nor;i++)
      {
            printf("%c: ",65+i);
            scanf("%d",&Av[i]);
      }
}
void acceptrequest()
{
      int i;
      printf("\nEnter the Process for which request has arrived :P");
      scanf("%d",&Rprocess);
      printf("\nEnter the request for process: ");
      for(i=0;i<nor;i++)
      {
            printf("%c: ",65+i);
            scanf("%d",&R[i]);
      }


}

void calcneed()
{
      int i,j;
      for(i=0;i<nop;i++)
            for(j=0;j<nor;j++)
                  N[i][j]=M[i][j]-A[i][j];

}
void displaydata()
{
      int i,j;
      printf("\n\tAllocation \t\tMax\t\tNeed\n\t");
      for(i=0;i<3;i++)
      {
            for(j=0;j<nor;j++)
                  printf("%4c",65+j);
            printf("\t");
      }
      for(i=0;i<nop;i++)
      {
            printf("\nP%d\t",i);
            for(j=0;j<nor;j++)
                  printf("%4d",A[i][j]);
            printf("\t");
            for(j=0;j<nor;j++)
                  printf("%4d",M[i][j]);
            printf("\t");
            for(j=0;j<nor;j++)
                  printf("%4d",N[i][j]);
      }
      printf("\navailable");
      for(i=0;i<nor;i++)
            printf("%4d",Av[i]);
}
int checkneed(int pno)
{
      int i;
      for(i=0;i<nor;i++)
            if(N[pno][i]>Av[i])
                  return 0;
      return 1;
}
void resourcerequest()
{
      int i;

      for(i=0;i<nor;i++)
      {
            if(R[i]>N[Rprocess][i])
                  break;
      }
      if(i==nor)
      {
            for(i=0;i<nor;i++)
            {
                  if(R[i]>Av[i])
                        break;
            }

      }
      if(i==nor)
      {
            printf("\nRequest<=Need \n Request<=Available \n Both
Condition are true");
            printf("\nThen system Pretends to fulfill request , then
modify resourse allocation state");
            for(i=0;i<nor;i++)
            {
                  Av[i]=Av[i]-R[i];
                  A[Rprocess][i]=A[Rprocess][i]+R[i];
                  N[Rprocess][i]=N[Rprocess][i]-R[i];
            }
            displaydata();
      }
      else
      {
            printf("\nRequest<=Need \n Request<=Available \n Condition is
not true");
            printf("\nSo request cannot be satisfied!");
      }

}

void banker()
{
      int p=0,j=0,k=0,flag=0,safe[10];
      while(flag<2)
      {
            if(!finish[p])
            {
                  printf("\n\nNeed of process P%d (,",p);
                  for(j=0;j<nor;j++)
                        printf("%d,",N[p][j]);
                  if(checkneed(p))
                  {
                        printf(") <= available (");
                        for(j=0;j<nor;j++)
                              printf("%d,",Av[j]);
                        printf(")");

                        printf("\nNeed is Satsified, So process P%d can be
granted requiered resources.\n After P%d finishes, it will realease all
the resources.",p,p);
                        for(j=0;j<nor;j++)
```

```
                              Av[j]=Av[j]+A[p][j];

                        printf("New Availble=");
                        for(j=0;j<nor;j++)
                              printf("%d ",Av[j]);
                        finish[p]=1;
                        safe[k++]=p;

                  }
                  else
                  {
                        printf(") >  available (");
                        for(j=0;j<nor;j++)
                              printf("%d,",Av[j]);
                        printf(")");

                        printf("\nNeed is not Satsified, So process P%d
cannot be granted required resources.\n process P%d has to wait.",p,p);
                  }
            }
            if((p+1)%nop==0)
                  flag++;
            p=(p+1)%nop;
      }//while
      if(k==nop)
      {
            printf("\nSystem is in safe state...");
            printf("\nSafe Sequence: ");
            for(j=0;j<k;j++)
                  printf("P%d->",safe[j]);
      }
      else
            printf("\nSystem is not in safe state....");
}


main()
{
      printf("\nEnter No of Processes: ");
      scanf("%d",&nop);
      printf("\nEnter No. of Resources: ");
      scanf("%d",&nor);
      printf("\nEnter Allocation Matrix: ");
      acceptdata(A);
      printf("\nEnter Max Matrix: ");
      acceptdata(M);
      printf("\nEnter Availability:");
      acceptav();
      calcneed();
      displaydata();
      acceptrequest();
      resourcerequest();
      banker();
}
/*
[root@localhost ~]# cc resoucerequest.c
```

```
[root@localhost ~]# ./a.out

Enter No of Processes: 5

Enter No. of Resources: 3

Enter Allocation Matrix: P0
A: 0
B: 1
C: 0
P1
A: 2
B: 0
C: 0
P2
A: 3
B: 0
C: 2
P3
A: 2
B: 1
C: 1
P4
A: 0
B: 0
C: 2

Enter Max Matrix: P0
A: 7
B: 5
C: 3
P1
A: 3
B: 2
C: 2
P2
A: 9
B: 0
C: 2
P3
A: 2
B: 2
C: 2
P4
A: 4
B: 3
C: 3

Enter Availability:A: 3
B: 3
C: 2

        Allocation              Max             Need
           A   B   C       A   B   C       A   B   C
P0         0   1   0       7   5   3       7   4   3
P1         2   0   0       3   2   2       1   2   2
```

```
P2         3   0   2       9   0   2       6   0   0
P3         2   1   1       2   2   2       0   1   1
P4         0   0   2       4   3   3       4   3   1
available   3   3   2
Enter the Process for which request has arrived :P1

Enter the request for process: A: 1
B: 0
C: 2

Request<=Need
 Request<=Available
 Both Condition are true
Then system Pretends to fulfill request , then modify resourse
allocation state
        Allocation              Max             Need
           A   B   C       A   B   C       A   B   C
P0         0   1   0       7   5   3       7   4   3
P1         3   0   2       3   2   2       0   2   0
P2         3   0   2       9   0   2       6   0   0
P3         2   1   1       2   2   2       0   1   1
P4         0   0   2       4   3   3       4   3   1
available   2   3   0

Need of process P0 (,7,4,3,) >  available (2,3,0,)
Need is not Satsified, So process P0 cannot be granted required
resources.
 process P0 has to wait.

Need of process P1 (,0,2,0,) <= available (2,3,0,)
Need is Satsified, So process P1 can be granted requiered resources.
 After P1 finishes, it will realease all the resources.New Availble=5 3
2

Need of process P2 (,6,0,0,) >  available (5,3,2,)
Need is not Satsified, So process P2 cannot be granted required
resources.
 process P2 has to wait.

Need of process P3 (,0,1,1,) <= available (5,3,2,)
Need is Satsified, So process P3 can be granted requiered resources.
 After P3 finishes, it will realease all the resources.New Availble=7 4
3

Need of process P4 (,4,3,1,) <= available (7,4,3,)
Need is Satsified, So process P4 can be granted requiered resources.
 After P4 finishes, it will realease all the resources.New Availble=7 4
5

Need of process P0 (,7,4,3,) <= available (7,4,5,)
Need is Satsified, So process P0 can be granted requiered resources.
 After P0 finishes, it will realease all the resources.New Availble=7 5
5

Need of process P2 (,6,0,0,) <= available (7,5,5,)
Need is Satsified, So process P2 can be granted requiered resources.
```

```
 After P2 finishes, it will realease all the resources.New Availble=10 5
7
System is in safe state...
Safe Sequence: P1->P3->P4->P0->P2->
*/
```