19BCE0829
KANDRA KSHEERAJ

CSE2005 - Operating Systems

LAB ASSIGNMENT-3
Slot: L35 + L36

Faculty: GERALDINE BESSIE AMALI D MAM

Date: 21-09-2020

19BCE0829 KSHEERAJ KANDRA

1. producer and consumer problem using Semaphore

Open *semaphore_demo.c Save

```c
#include<stdio.h>
#include<sys/types.h>
#include<pthread.h>
#include<semaphore.h>
sem_t mutex;
sem_t full;
sem_t empty;
char buffer[10];

void *producer()
{
int i,index=0;
for(i=0;i<26;i++)
{
sem_wait(&empty);//looping
sem_wait(&mutex);
buffer[index]=i+65;
printf("producer added %c to buffer\n",buffer[index]);
sem_post(&full);
sem_post(&mutex);
if(++index==10)
index=0;
}
}
void  *consumer()
{
int i,index= 0;
for(i=0;i<26;i++)
```

```
{
sem_wait(&full);//looping
sem_wait(&mutex);
printf("consumer consumed %c\n",buffer[index]);
sem_post(&empty);
sem_post(&mutex);
if(++index==10)
index=0;
}
}

int main()
{
pthread_t tid1,tid2,tid3,tid4;
sem_init(&mutex,0,1);
sem_init(&full,0,0);
sem_init(&empty,0,10);
pthread_create(&tid1,NULL,producer,NULL);
pthread_create(&tid2,NULL,consumer,NULL);
//pthread_create(&tid3,NULL,producer,NULL);
//pthread_create(&tid4,NULL,producer,NULL);
pthread_join(tid1,NULL);
pthread_join(tid2,NULL);
sem_destroy(&mutex);
sem_destroy(&full);
sem_destroy(&empty);
return 0;
}
```

19BCE0829
KANDRA KSHEERAJ

```
kandraksheeraj@srikithadesk-VirtualBox: ~
kandraksheeraj@srikithadesk-VirtualBox:~$ gedit semaphore_demo.c
kandraksheeraj@srikithadesk-VirtualBox:~$ gcc semaphore_demo.c -lpthread
kandraksheeraj@srikithadesk-VirtualBox:~$ ./a.out
producer added A to buffer
producer added B to buffer
producer added C to buffer
producer added D to buffer
producer added E to buffer
producer added F to buffer
producer added G to buffer
producer added H to buffer
producer added I to buffer
producer added J to buffer
consumer consumed A
consumer consumed B
consumer consumed C
consumer consumed D
consumer consumed E
consumer consumed F
consumer consumed G
consumer consumed H
consumer consumed I
consumer consumed J
producer added K to buffer
producer added L to buffer
producer added M to buffer
producer added N to buffer
producer added O to buffer
producer added P to buffer
```

```
producer added Q to buffer
producer added R to buffer
producer added S to buffer
producer added T to buffer
consumer consumed K
consumer consumed L
consumer consumed M
consumer consumed N
consumer consumed O
consumer consumed P
consumer consumed Q
consumer consumed R
consumer consumed S
consumer consumed T
producer added U to buffer
producer added V to buffer
producer added W to buffer
producer added X to buffer
producer added Y to buffer
producer added Z to buffer
consumer consumed U
consumer consumed V
consumer consumed W
consumer consumed X
consumer consumed Y
consumer consumed Z
kandraksheeraj@srikithadesk-VirtualBox:~$
```

2. Multiple producers and consumers problem using Semaphore with Threads

semaphore_multidemo.c

```c
#include<stdio.h>
#include<sys/types.h>
#include<pthread.h>
#include<semaphore.h>
sem_t mutex;
sem_t full;
sem_t empty;
char buffer[10];

void *producer(void *arg)
{
int i,index=0;
for(i=0;i<26;i++)
{
sem_wait(&empty);//looping
sem_wait(&mutex);
buffer[index]=i+65;
printf("producer added %c to buffer\t with thread id %ld
\n",buffer[index],pthread_self());
sem_post(&full);
sem_post(&mutex);
if(++index==10)
index=0;
if(rand()%5==0)
sleep(1);
}
}
void  *consumer()
```

```
{
int i,index= 0;
for(i=0;i<26;i++)
{
sem_wait(&full);//looping
sem_wait(&mutex);
printf("consumer consumed %c\t with thread id %ld
\n",buffer[index],pthread_self());
sem_post(&empty);
sem_post(&mutex);
if(++index==10)
index=0;
if(rand()%3==0)
sleep(2);
}
}

int main()
{
pthread_t tid1[10],tid2[10];
sem_init(&mutex,0,1);
sem_init(&full,0,0);
sem_init(&empty,0,10);
int i;
for(i=0;i<10;i++)
{
```

```
pthread_create(&tid1[i],NULL,producer,NULL);
pthread_create(&tid2[i],NULL,consumer,NULL);
}
//pthread_create(&tid3,NULL,producer,NULL);
//pthread_create(&tid4,NULL,producer,NULL);
for(i=0;i<10;i++)
{
pthread_join(tid1[i],NULL);
pthread_join(tid2[i],NULL);
}
sem_destroy(&mutex);
sem_destroy(&full);
sem_destroy(&empty);
return 0;
}
```

19BCE0829
KANDRA KSHEERAJ

```
kandraksheeraj@srikithadesk-VirtualBox: ~
kandraksheeraj@srikithadesk-VirtualBox:~$ gedit semaphore_multidemo.c
kandraksheeraj@srikithadesk-VirtualBox:~$ gcc semaphore_multidemo.c -lpthread
semaphore_multidemo.c: In function 'producer':
semaphore_multidemo.c:23:4: warning: implicit declaration of function 'rand' [-
Wimplicit-function-declaration]
   23 | if(rand()%5==0)
      |    ^~~~
semaphore_multidemo.c:24:1: warning: implicit declaration of function 'sleep' [
-Wimplicit-function-declaration]
   24 | sleep(1);
      | ^~~~~
kandraksheeraj@srikithadesk-VirtualBox:~$ ./a.out
producer added A to buffer       with thread id 140372643346176
producer added B to buffer       with thread id 140372643346176
producer added C to buffer       with thread id 140372643346176
producer added D to buffer       with thread id 140372643346176
producer added A to buffer       with thread id 140372609775360
producer added B to buffer       with thread id 140372609775360
consumer consumed A      with thread id 140372634953472
consumer consumed B      with thread id 140372634953472
producer added A to buffer       with thread id 140372592989952
producer added B to buffer       with thread id 140372592989952
consumer consumed A      with thread id 140372618168064
consumer consumed B      with thread id 140372618168064
consumer consumed A      with thread id 140372601382656
consumer consumed A      with thread id 140372651738880
consumer consumed A      with thread id 140372668524288
producer added A to buffer       with thread id 140372626560768
producer added B to buffer       with thread id 140372626560768
```

```
producer added B to buffer       with thread id 140372626560768
consumer consumed A      with thread id 140372584597248
producer added A to buffer       with thread id 140372559419136
producer added B to buffer       with thread id 140372559419136
producer added C to buffer       with thread id 140372559419136
consumer consumed B      with thread id 140372651738880
consumer consumed A      with thread id 140372551026432
producer added A to buffer       with thread id 140372576204544
producer added B to buffer       with thread id 140372576204544
producer added C to buffer       with thread id 140372576204544
producer added A to buffer       with thread id 140372660131584
consumer consumed C      with thread id 140372651738880
consumer consumed D      with thread id 140372651738880
consumer consumed B      with thread id 140372601382656
producer added A to buffer       with thread id 140372542633728
producer added B to buffer       with thread id 140372542633728
producer added C to buffer       with thread id 140372542633728
consumer consumed A      with thread id 140372567811840
producer added A to buffer       with thread id 140372525848320
producer added B to buffer       with thread id 140372525848320
producer added B to buffer       with thread id 140372660131584
consumer consumed A      with thread id 140372534241024
consumer consumed B      with thread id 140372534241024
consumer consumed C      with thread id 140372534241024
consumer consumed D      with thread id 140372534241024
producer added C to buffer       with thread id 140372592989952
producer added D to buffer       with thread id 140372592989952
consumer consumed A      with thread id 140372517455616
consumer consumed B      with thread id 140372517455616
```

19BCE0829
KANDRA KSHEERAJ

```
consumer consumed C      with thread id 140372517455616
consumer consumed B      with thread id 140372668524288
producer added C to buffer       with thread id 140372525848320
producer added D to buffer       with thread id 140372525848320
consumer consumed C      with thread id 140372668524288
consumer consumed C      with thread id 140372618168064
consumer consumed D      with thread id 140372618168064
consumer consumed        with thread id 140372651738880
producer added C to buffer       with thread id 140372660131584
consumer consumed        with thread id 140372618168064
consumer consumed C      with thread id 140372601382656
producer added E to buffer       with thread id 140372592989952
producer added F to buffer       with thread id 140372592989952
consumer consumed F      with thread id 140372651738880
producer added D to buffer       with thread id 140372559419136
producer added E to buffer       with thread id 140372559419136
consumer consumed F      with thread id 140372618168064
consumer consumed        with thread id 140372651738880
consumer consumed        with thread id 140372651738880
producer added D to buffer       with thread id 140372542633728
producer added E to buffer       with thread id 140372542633728
consumer consumed D      with thread id 140372517455616
consumer consumed E      with thread id 140372517455616
producer added A to buffer       with thread id 140372676916992
producer added B to buffer       with thread id 140372676916992
producer added C to buffer       with thread id 140372676916992
producer added D to buffer       with thread id 140372676916992
producer added C to buffer       with thread id 140372609775360
```

```
producer added C to buffer       with thread id 140372609775360
producer added D to buffer       with thread id 140372609775360
producer added E to buffer       with thread id 140372643346176
producer added F to buffer       with thread id 140372643346176
producer added G to buffer       with thread id 140372643346176
producer added H to buffer       with thread id 140372643346176
consumer consumed C      with thread id 140372634953472
producer added I to buffer       with thread id 140372643346176
consumer consumed B      with thread id 140372584597248
consumer consumed C      with thread id 140372584597248
producer added C to buffer       with thread id 140372626560768
producer added D to buffer       with thread id 140372626560768
consumer consumed B      with thread id 140372551026432
consumer consumed C      with thread id 140372551026432
producer added D to buffer       with thread id 140372660131584
producer added E to buffer       with thread id 140372660131584
consumer consumed B      with thread id 140372567811840
consumer consumed C      with thread id 140372567811840
consumer consumed D      with thread id 140372567811840
producer added F to buffer       with thread id 140372559419136
producer added G to buffer       with thread id 140372592989952
producer added H to buffer       with thread id 140372592989952
consumer consumed D      with thread id 140372668524288
consumer consumed E      with thread id 140372668524288
consumer consumed F      with thread id 140372668524288
consumer consumed G      with thread id 140372668524288
producer added E to buffer       with thread id 140372626560768
producer added F to buffer       with thread id 140372626560768
consumer consumed G      with thread id 140372618168064
```

```
consumer consumed D      with thread id 140372601382656
consumer consumed E      with thread id 140372601382656
consumer consumed F      with thread id 140372601382656
consumer consumed G      with thread id 140372601382656
producer added D to buffer       with thread id 140372576204544
producer added E to buffer       with thread id 140372576204544
producer added F to buffer       with thread id 140372660131584
consumer consumed I      with thread id 140372651738880
producer added E to buffer       with thread id 140372525848320
producer added F to buffer       with thread id 140372525848320
producer added I to buffer       with thread id 140372592989952
consumer consumed E      with thread id 140372534241024
consumer consumed F      with thread id 140372534241024
consumer consumed G      with thread id 140372534241024
consumer consumed H      with thread id 140372534241024
consumer consumed I      with thread id 140372534241024
producer added G to buffer       with thread id 140372626560768
producer added H to buffer       with thread id 140372626560768
producer added F to buffer       with thread id 140372576204544
producer added E to buffer       with thread id 140372609775360
producer added G to buffer       with thread id 140372660131584
producer added F to buffer       with thread id 140372542633728
producer added E to buffer       with thread id 140372676916992
consumer consumed H      with thread id 140372601382656
consumer consumed F      with thread id 140372517455616
consumer consumed G      with thread id 140372517455616
producer added J to buffer       with thread id 140372592989952
```

…….. so on to Z producers and consumers

```
consumer consumed V      with thread id 140372567811840
producer added W to buffer       with thread id 140372576204544
consumer consumed T      with thread id 140372534241024
producer added V to buffer       with thread id 140372626560768
consumer consumed X      with thread id 140372634953472
producer added X to buffer       with thread id 140372576204544
consumer consumed W      with thread id 140372551026432
consumer consumed X      with thread id 140372551026432
consumer consumed Y      with thread id 140372551026432
consumer consumed Z      with thread id 140372551026432
producer added W to buffer       with thread id 140372626560768
producer added Y to buffer       with thread id 140372576204544
producer added Z to buffer       with thread id 140372576204544
consumer consumed W      with thread id 140372567811840
consumer consumed X      with thread id 140372567811840
consumer consumed U      with thread id 140372534241024
producer added X to buffer       with thread id 140372626560768
consumer consumed Y      with thread id 140372634953472
producer added Y to buffer       with thread id 140372626560768
producer added Z to buffer       with thread id 140372626560768
consumer consumed Y      with thread id 140372567811840
consumer consumed Z      with thread id 140372567811840
consumer consumed V      with thread id 140372534241024
consumer consumed Z      with thread id 140372634953472
consumer consumed W      with thread id 140372534241024
consumer consumed X      with thread id 140372534241024
consumer consumed Y      with thread id 140372534241024
consumer consumed Z      with thread id 140372534241024
kandraksheeraj@srikithadesk-VirtualBox:~$
```

19BCE0829
KANDRA KSHEERAJ

3. Readers writers problem

Open read_writers.c Save

```c
#include<semaphore.h>
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<pthread.h>
sem_t x,y;
pthread_t tid;
pthread_t writerthreads[100],readerthreads[100];
int readercount = 0;

void *reader(void* param)
{
    sem_wait(&x);
    readercount++;
    if(readercount==1)
        sem_wait(&y);
    sem_post(&x);
    printf("%d reader is inside\n",readercount);
    usleep(3);
    sem_wait(&x);
    readercount--;
```

```c
    if(readercount==0)
    {
        sem_post(&y);
    }
    sem_post(&x);
    printf("%d Reader is leaving\n",readercount+1);
    return NULL;
}
void *writer(void* param)
{
    printf("Writer is trying to enter\n");
    sem_wait(&y);
    printf("Writer has entered\n");
    sem_post(&y);
    printf("Writer is leaving\n");
    return NULL;
}

```

19BCE0829
KANDRA KSHEERAJ

```c
int main()
{
    int n2,i;
    printf("Enter the number of readers:");
    scanf("%d",&n2);
    printf("\n");
    int n1[n2];
    sem_init(&x,0,1);
    sem_init(&y,0,1);
    for(i=0;i<n2;i++)
    {
        pthread_create(&writerthreads[i],NULL,reader,NULL);
        pthread_create(&readerthreads[i],NULL,writer,NULL);
    }
    for(i=0;i<n2;i++)
    {
        pthread_join(writerthreads[i],NULL);
        pthread_join(readerthreads[i],NULL);
    }

}

```

```
kandraksheeraj@srikithadesk-VirtualBox: ~
kandraksheeraj@srikithadesk-VirtualBox:~$ gedit read_writers.c
kandraksheeraj@srikithadesk-VirtualBox:~$ gcc read_writers.c -lpthread
kandraksheeraj@srikithadesk-VirtualBox:~$ ./a.out
Enter the number of readers:7

1 reader is inside
Writer is trying to enter
Writer is trying to enter
2 reader is inside
3 reader is inside
Writer is trying to enter
Writer is trying to enter
4 reader is inside
Writer is trying to enter
5 reader is inside
Writer is trying to enter
6 reader is inside
Writer is trying to enter
7 reader is inside
7 Reader is leaving
6 Reader is leaving
5 Reader is leaving
4 Reader is leaving
3 Reader is leaving
2 Reader is leaving
1 Reader is leaving
Writer has entered
Writer is leaving
Writer has entered
```

19BCE0829
KANDRA KSHEERAJ

```
4 reader is inside
Writer is trying to enter
5 reader is inside
Writer is trying to enter
6 reader is inside
Writer is trying to enter
7 reader is inside
7 Reader is leaving
6 Reader is leaving
5 Reader is leaving
4 Reader is leaving
3 Reader is leaving
2 Reader is leaving
1 Reader is leaving
Writer has entered
Writer is leaving
Writer has entered
Writer is leaving
Writer has entered
Writer is leaving
Writer has entered
Writer is leaving
Writer has entered
Writer is leaving
Writer has entered
Writer is leaving
Writer has entered
Writer is leaving
kandraksheeraj@srikithadesk-VirtualBox:~$
```

19BCE0829
KANDRA KSHEERAJ

4. Dining philosophers problem

Open din_phi.c Save

```c
#include <pthread.h>
#include <semaphore.h>
#include<stdio.h>

#define n 4

int compltedPhilo = 0,i;

struct fork{
        int taken;
}ForkAvil[n];

struct philosp{
        int left;
        int right;
}Philostatus[n];

void goForDinner(int philID){
        if(Philostatus[philID].left==10 && Philostatus[philID].right==10)
        printf("Philosopher %d completed his dinner\n",philID+1);

        else if(Philostatus[philID].left==1 && Philostatus[philID].right==1)
{

            printf("Philosopher %d completed his dinner\n",philID+1);

            Philostatus[philID].left = Philostatus[philID].right = 10;
            int otherFork = philID-1;
```

```c

            if(otherFork== -1)
                otherFork=(n-1);

            ForkAvil[philID].taken = ForkAvil[otherFork].taken = 0;
            printf("Philosopher %d released fork %d and fork %d\n",philID+1,philID+1,otherFork+1);
            compltedPhilo++;
        }
        else if(Philostatus[philID].left==1 && Philostatus[philID].right==0){
                if(philID==(n-1)){
                    if(ForkAvil[philID].taken==0){
                        ForkAvil[philID].taken = Philostatus[philID].right = 1;
                        printf("Fork %d taken by philosopher %d\n",philID+1,philID+1);
                    }else{
                        printf("Philosopher %d is waiting for fork %d\n",philID+1,philID+1);
                    }
                }else{
                    int dupphilID = philID;
                    philID-=1;

                    if(philID== -1)
                        philID=(n-1);

```

```c
                    if(ForkAvil[philID].taken == 0){
                        ForkAvil[philID].taken = Philostatus[dupphilID].right = 1;
                        printf("Fork %d taken by Philosopher %d\n",philID+1,dupphilID+1);
                    }else{
                        printf("Philosopher %d is waiting for Fork %d\n",dupphilID+1,philID+1);
                    }
                }
            }
            else if(Philostatus[philID].left==0){
                    if(philID==(n-1)){
                        if(ForkAvil[philID-1].taken==0){
                            ForkAvil[philID-1].taken = Philostatus[philID].left = 1;
                            printf("Fork %d taken by philosopher %d\n",philID,philID+1);
                        }else{
                            printf("Philosopher %d is waiting for fork %d\n",philID+1,philID);
                        }
                    }else{
                        if(ForkAvil[philID].taken == 0){
                            ForkAvil[philID].taken = Philostatus[philID].left = 1;
```

19BCE0829
KANDRA KSHEERAJ

```c
                                printf("Fork %d taken by Philosopher %d\n",philID+1,philID+1);
                            }else{
                                printf("Philosopher %d is waiting for Fork %d\n",philID+1,philID+1);
                            }
                        }
        }else{}
}

int main(){
        for(i=0;i<n;i++)
        ForkAvil[i].taken=Philostatus[i].left=Philostatus[i].right=0;

        while(compltedPhilo<n){

                for(i=0;i<n;i++)
            goForDinner(i);
                printf("\nTill now num of philosophers completed dinner are %d\n\n",compltedPhilo);
        }

        return 0;
}
```

```
kandraksheeraj@srikithadesk-VirtualBox: ~
kandraksheeraj@srikithadesk-VirtualBox:~$ gedit din_phi.c
kandraksheeraj@srikithadesk-VirtualBox:~$ gcc din_phi.c
kandraksheeraj@srikithadesk-VirtualBox:~$ ./a.out
Fork 1 taken by Philosopher 1
Fork 2 taken by Philosopher 2
Fork 3 taken by Philosopher 3
Philosopher 4 is waiting for fork 3

Till now num of philosophers completed dinner are 0

Fork 4 taken by Philosopher 1
Philosopher 2 is waiting for Fork 1
Philosopher 3 is waiting for Fork 2
Philosopher 4 is waiting for fork 3

Till now num of philosophers completed dinner are 0

Philosopher 1 completed his dinner
Philosopher 1 released fork 1 and fork 4
Fork 1 taken by Philosopher 2
Philosopher 3 is waiting for Fork 2
Philosopher 4 is waiting for fork 3
```

19BCE0829
KANDRA KSHEERAJ

```
Till now num of philosophers completed dinner are 1

Philosopher 1 completed his dinner
Philosopher 2 completed his dinner
Philosopher 2 released fork 2 and fork 1
Fork 2 taken by Philosopher 3
Philosopher 4 is waiting for fork 3

Till now num of philosophers completed dinner are 2

Philosopher 1 completed his dinner
Philosopher 2 completed his dinner
Philosopher 3 completed his dinner
Philosopher 3 released fork 3 and fork 2
Fork 3 taken by philosopher 4
```

```
Till now num of philosophers completed dinner are 3

Philosopher 1 completed his dinner
Philosopher 2 completed his dinner
Philosopher 3 completed his dinner
Fork 4 taken by philosopher 4

Till now num of philosophers completed dinner are 3

Philosopher 1 completed his dinner
Philosopher 2 completed his dinner
Philosopher 3 completed his dinner
Philosopher 4 completed his dinner
Philosopher 4 released fork 4 and fork 3

Till now num of philosophers completed dinner are 4

kandraksheeraj@srikithadesk-VirtualBox:~$
```

19BCE0829
KANDRA KSHEERAJ

5. Program to avoid deadlock using Banker's algorithm (Safety algorithm)

Open *bank_alg.c Save

```c
#include<stdio.h>

void Need(int n,int m,int allo[n][m],int max[n][m],int need[n][m]){
                for(int i=0;i<n;i++){
                for(int j=0;j<m;j++){
                need[i][j]=max[i][j]-allo[i][j];
                }
                }
}
void display(int safeseq[],int n){
        printf("\nSafe Sequence is:\n");
        for(int i=0;i<n;i++){
        printf(" --> process %d" ,safeseq[i]);
        }
        printf("\n");
}
void Safeseq(int n,int m,int need[n][m],int alloc[n][m],int avail[m],int finish[n],int safeseq[n]){
                        int i=0,I=0;
                        while(i<n){
                        if(finish[i]==0){
                        int c=0;
                        for(int j=0;j<m;j++){
                        if(need[i][j]<=avail[j]){
                        c+=1 ;
                        }
                        }
                        if(c==m){
```

```
                        for(int j=0;j<m;j++){
                        avail[j]+=alloc[i][j];
                        }
                        finish[i]=1;
                        safeseq[I]=i;
                        I++;
                        }
                        i++;
                        if(i>=n){
                                i-=n;
                        }
                        }
                        else{
                                i++;
                                continue;
                        }
                        }
}
int main(){
        int n,m;
        printf("\nEnter number of processes: ");
        scanf("%d",&n);
        printf("\nEnter number of resources available: ");
        scanf("%d",&m);

        int avail[m];
        for(int i=0;i<m;i++){
                printf("\nEnter Available instances of Resource %d: ",i+1);
                scanf("%d",&avail[i]);
        }
        int max[n][m];
        for(int i=0;i<n;i++){
                printf("\nEnter max resource demand of process %d: ",i);
        for(int j=0;j<m;j++){
                printf("\n\tEnter demand of resource %d: ",j+1);
                scanf("%d",&max[i][j]);
        }
        }
```

```c
        int alloc[n][m];
        for(int i=0;i<n;i++){
                printf("\nEnter current resource allocated for process %d: ",i);
        for(int j=0;j<m;j++){
                printf("\n\tEnter allocated resource %d: ",j+1);
                scanf("%d",&alloc[i][j]);
        }
        }

        int need[n][m];
        Need(n,m,alloc,max,need);

        int finish[n];
        for(int i=0;i<n;i++){
                finish[i]=0;
        }

        int safeseq[n];
        Safeseq(n,m,need,alloc,avail,finish,safeseq);
        display(safeseq,n);
        return 0;
}
```

```
kandraksheeraj@srikithadesk-VirtualBox: ~
kandraksheeraj@srikithadesk-VirtualBox:~$ gcc bank_alg.c
kandraksheeraj@srikithadesk-VirtualBox:~$ ./a.out

Enter number of processes: 5

Enter number of resources available: 3

Enter Available instances of Resource 1: 3

Enter Available instances of Resource 2: 3

Enter Available instances of Resource 3: 2

Enter max resource demand of process 0:
        Enter demand of resource 1: 7

        Enter demand of resource 2: 5

        Enter demand of resource 3: 3

Enter max resource demand of process 1:
        Enter demand of resource 1: 3

        Enter demand of resource 2: 2

        Enter demand of resource 3: 2
```

```
Enter max resource demand of process 2:
        Enter demand of resource 1: 9

        Enter demand of resource 2: 0

        Enter demand of resource 3: 2

Enter max resource demand of process 3:
        Enter demand of resource 1: 2

        Enter demand of resource 2: 2

        Enter demand of resource 3: 2

Enter max resource demand of process 4:
        Enter demand of resource 1: 4

        Enter demand of resource 2: 3

        Enter demand of resource 3: 3

Enter current resource allocated for process 0:
        Enter allocated resource 1: 0

        Enter allocated resource 2: 1

        Enter allocated resource 3: 0
```

```
Enter current resource allocated for process 1:
        Enter allocated resource 1: 2

        Enter allocated resource 2: 0

        Enter allocated resource 3: 0

Enter current resource allocated for process 2:
        Enter allocated resource 1: 3

        Enter allocated resource 2: 0

        Enter allocated resource 3: 2
```

```
Enter current resource allocated for process 3:
        Enter allocated resource 1: 2

        Enter allocated resource 2: 2

        Enter allocated resource 3: 1

Enter current resource allocated for process 4:
        Enter allocated resource 1: 0

        Enter allocated resource 2: 0

        Enter allocated resource 3: 2

Safe Sequence is:
 --> process 1 --> process 3 --> process 4 --> process 0 --> process 2
kandraksheeraj@srikithadesk-VirtualBox:~$
```

6. Program to avoid deadlock using Banker's algorithm (including Safe state and additional resource request)

bank_alg2.c

```c
#include<stdio.h>

void function(int n,int m){
        int flag;
        struct p{
                int max[m],alloc[m],need[m];
                }p[n];
        int avail[m], request[m];

        for(int i=0;i<m;i++){
                printf("\nEnter Available instances of Resource %d: ",i+1);
                scanf("%d",&avail[i]);
        }
        printf("\n");
        for(int i=0;i<n;i++){
                printf("\nEnter Details of process %d:",i);
        for(int j=0;j<m;j++){
                printf("\nEnter max resource (%d) required;",j+1);
        scanf("%d",&p[i].max[j]);
        }

        for(int j=0;j<m;j++){
                printf("\n\tEnter allocated resourse (%d):",j+1);
                scanf("%d",&p[i].alloc[j]);
        }

        printf("\n\tDoes this process nedd additional resource?\(Enter 1 for Yes 0 for No)");
```

```
        scanf("%d",&flag);

        if(flag==0){
                for(int j=0;j<m;j++){
                        p[i].need[j]=p[i].max[j]-p[i].alloc[j];
                }
                }
                else{
                        printf("\nEnter Request Details:");
                for(int j=0;j<m;j++){
                        printf("\n\tEnter request of resource %d",j+1);
                        scanf("%d",&request[j]);
                        avail[j]-=request[j];
                }

                for(int j=0;j<m;j++){
                        p[i].alloc[j]+=request[j];
                        p[i].need[j]=p[i].max[j]-p[i].alloc[j];
                }
                }
            }

            int finish[n];
            for(int i=0;i<n;i++){
                finish[i]=0;
                }

                int safeseq[n];
```

```
                int i=0,I=0;
                while(i<n){
                        if(finish[i]==0){
                                int c=0;
                        for(int j=0;j<m;j++){
                                if(p[i].need[j]<=avail[j]){
                            c+=1;
                                }
                        }

                        if(c==m){
                                for(int j=0;j<m;j++){
                                        avail[j]+=p[i].alloc[j];
                                }
                                finish[i]=1;
                                safeseq[I]=i;
                                I++;
                        }
                        i++;
                        if(i>=n){
                                i-=n;
                        }
                        }
                        else{
                                i++;
                                continue;
                        }
                }
```

```c

                printf("\nSafe Sequence is:\n");
                for(int i=0;i<n;i++){
                        printf(" -->process %d" ,safeseq[i]);
                }
                printf("\n");
}


int main(){
        int n,m;
        printf("Enter number of processes:\n");
        scanf("%d",&n);
        printf("Enter number of resources:\n");
        scanf("%d",&m);

        function(n,m);

        return 0;
}
```

```
kandraksheeraj@srikithadesk-VirtualBox: ~
kandraksheeraj@srikithadesk-VirtualBox:~$ gcc bank_alg2.c
kandraksheeraj@srikithadesk-VirtualBox:~$ ./a.out
Enter number of processes:
5
Enter number of resources:
3

Enter Available instances of Resource 1: 3

Enter Available instances of Resource 2: 3

Enter Available instances of Resource 3: 2


Enter Details of process 0:
Enter max resource (1) required;7

Enter max resource (2) required;5

Enter max resource (3) required;3

        Enter allocated resourse (1):0

        Enter allocated resourse (2):1

        Enter allocated resourse (3):0

        Does this process nedd additional resource?(Enter 1 for Yes 0 for No)0
```

19BCE0829
KANDRA KSHEERAJ

```
Enter Details of process 1:
Enter max resource (1) required;3

Enter max resource (2) required;2

Enter max resource (3) required;2

        Enter allocated resourse (1):3

        Enter allocated resourse (2):3

        Enter allocated resourse (3):2

        Does this process nedd additional resource?(Enter 1 for Yes 0 for No)1

Enter Request Details:
        Enter request of resource 11

        Enter request of resource 20

        Enter request of resource 32

Enter Details of process 2:
Enter max resource (1) required;9

Enter max resource (2) required;0

Enter max resource (3) required;2
```

```
        Enter allocated resourse (1):3

        Enter allocated resourse (2):0

        Enter allocated resourse (3):2

        Does this process nedd additional resource?(Enter 1 for Yes 0 for No)0

Enter Details of process 3:
Enter max resource (1) required;2

Enter max resource (2) required;2

Enter max resource (3) required;2

        Enter allocated resourse (1):2

        Enter allocated resourse (2):1

        Enter allocated resourse (3):1

        Does this process nedd additional resource?(Enter 1 for Yes 0 for No)0

Enter Details of process 4:
Enter max resource (1) required;4

Enter max resource (2) required;3

Enter max resource (3) required;3
```

19BCE0829
KANDRA KSHEERAJ

```
Enter max resource (3) required;2

        Enter allocated resourse (1):2

        Enter allocated resourse (2):1

        Enter allocated resourse (3):1

        Does this process nedd additional resource?(Enter 1 for Yes 0 for No)0

Enter Details of process 4:
Enter max resource (1) required;4

Enter max resource (2) required;3

Enter max resource (3) required;3

        Enter allocated resourse (1):0

        Enter allocated resourse (2):0

        Enter allocated resourse (3):2

        Does this process nedd additional resource?(Enter 1 for Yes 0 for No)0

Safe Sequence is:
 -->process 1 -->process 2 -->process 3 -->process 4 -->process 0
kandraksheeraj@srikithadesk-VirtualBox:~$
```
