NAME-SHREYAS NEHE
REG.NO-21BCE3359

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

GENERAL SEMESTER 2023-24

B.Tech - CSE

BCSE303P: Operating Systems Lab

DIGITAL LAB ASSIGNMENT -3

NAME-SHREYAS NEHE

REG.NO-21BCE3359

SLOT-L49+L50

Q3A Implement process synchronization using semaphores.

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

#define BUFFER_SIZE 5

sem_t empty;
sem_t full;
int buffer[BUFFER_SIZE];
int index = 0;

void* producer(void* arg) {
    int item = *((int*)arg);
    sem_wait(&empty);
```

```
    // Critical section
    buffer[index] = item;
    printf("Produced item: %d\n", item);
    index++;

    sem_post(&full);
    pthread_exit(NULL);
}

void* consumer(void* arg) {
    sem_wait(&full);

    // Critical section
    int item = buffer[index - 1];
    printf("Consumed item: %d\n", item);
    index--;

    sem_post(&empty);
    pthread_exit(NULL);
}

int main() {
    int numItems;
    printf("Enter the number of items to produce and consume: ");
    scanf("%d", &numItems);

    sem_init(&empty, 0, BUFFER_SIZE); // Initialize empty semaphore
with BUFFER_SIZE
    sem_init(&full, 0, 0); // Initialize full semaphore with 0

    pthread_t producerThread, consumerThread;
```

```
    for (int i = 1; i <= numItems; i++) {
        pthread_create(&producerThread, NULL, producer, &i);
        pthread_create(&consumerThread, NULL, consumer, NULL);
        pthread_join(producerThread, NULL);
        pthread_join(consumerThread, NULL);
    }

    sem_destroy(&empty);
    sem_destroy(&full);

    return 0;
}
```

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

#define BUFFER_SIZE 5

sem_t empty;
sem_t full;
int buffer[BUFFER_SIZE];
int index = 0;

void* producer(void* arg) {
    int item = *((int*)arg);
    sem_wait(&empty);

    // Critical section
    buffer[index] = item;
    printf("Produced item: %d\n", item);
    index++;

    sem_post(&full);
    pthread_exit(NULL);
}

void* consumer(void* arg) {
    sem_wait(&full);

    // Critical section
    int item = buffer[index - 1];
    printf("Consumed item: %d\n", item);
    index--;

    sem_post(&empty);
    pthread_exit(NULL);
}
```

NAME-SHREYAS NEHE
REG.NO-21BCE3359

```c
void* threadFunction(void* threadId) {
    int tid = *((int*)threadId);

    printf("Thread %d is waiting.\n", tid);
    sem_wait(&semaphore);
    printf("Thread %d has acquired the semaphore.\n", tid);

    // Critical section
    printf("Thread %d is in the critical section.\n", tid);

    printf("Thread %d is releasing the semaphore.\n", tid);
    sem_post(&semaphore);

    pthread_exit(NULL);
}
```

```c
int main() {
    int numItems;
    printf("Enter the number of items to produce and consume: ");
    scanf("%d", &numItems);

    sem_init(&empty, 0, BUFFER_SIZE); // Initialize empty semaphore with BUFFER_SIZE
    sem_init(&full, 0, 0); // Initialize full semaphore with 0

    pthread_t producerThread, consumerThread;

    for (int i = 1; i <= numItems; i++) {
        pthread_create(&producerThread, NULL, producer, &i);
        pthread_create(&consumerThread, NULL, consumer, NULL);
        pthread_join(producerThread, NULL);
        pthread_join(consumerThread, NULL);
    }

    sem_destroy(&empty);
    sem_destroy(&full);

    return 0;
}
```

NAME-SHREYAS NEHE
REG.NO-21BCE3359

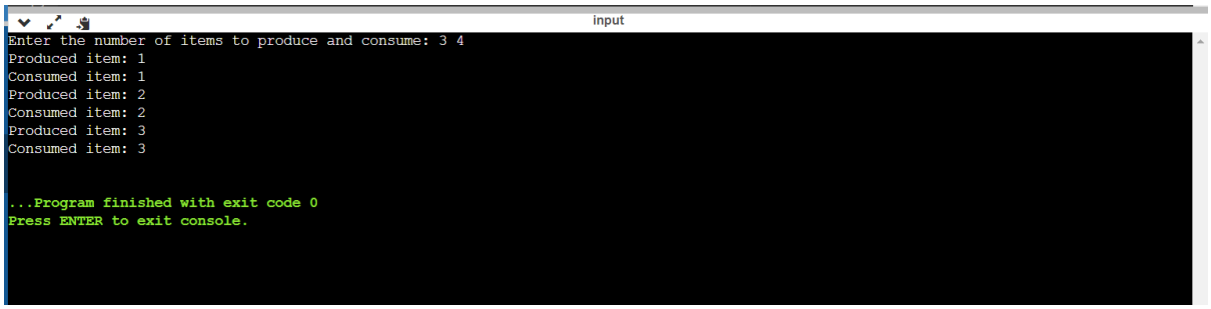
OUTPUT:

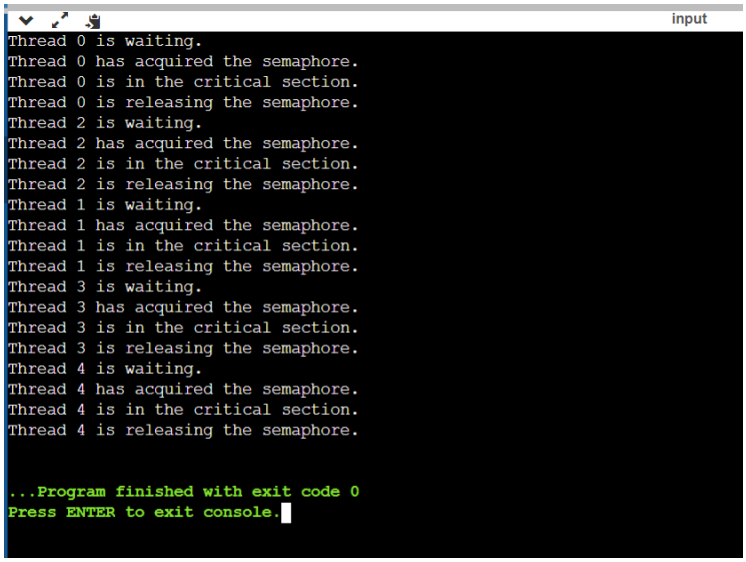
```
Enter the number of items to produce and consume: 2 2
Produced item: 1
Consumed item: 1
Produced item: 2
Consumed item: 2


...Program finished with exit code 0
Press ENTER to exit console.
```

input

```
Thread 0 is waiting.
Thread 0 has acquired the semaphore.
Thread 0 is in the critical section.
Thread 0 is releasing the semaphore.
Thread 1 is waiting.
Thread 1 has acquired the semaphore.
Thread 1 is in the critical section.
Thread 1 is releasing the semaphore.
Thread 2 is waiting.
Thread 2 has acquired the semaphore.
Thread 2 is in the critical section.
Thread 2 is releasing the semaphore.
Thread 4 is waiting.
Thread 4 has acquired the semaphore.
Thread 4 is in the critical section.
Thread 4 is releasing the semaphore.
Thread 3 is waiting.
Thread 3 has acquired the semaphore.
Thread 3 is in the critical section.
Thread 3 is releasing the semaphore.


...Program finished with exit code 0
Press ENTER to exit console.
```

NAME-SHREYAS NEHE
REG.NO-21BCE3359





NAME-SHREYAS NEHE
REG.NO-21BCE3359

Q3B Simulation of Banker s algorithm to check whether the given system is in safe state or not. Also check whether addition resource requested can be granted immediately.

CODE:

```
#include<stdio.h>


    int n,m;
    int i,j;
    int a[100][100];
    int highest[100][100];
    int whichisavailablee[100];
     int needd[100][100];
     int w[100];
     int f[100];
     int safseq[100];


     void init()
     {
         int i,j;
         for(i=0;i<m;i++)
         w[i]=whichisavailablee[i];


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


         for(i=0;i<n;i++)
         {
             for(j=0;j<m;j++)
             needd[i][j]=highest[i][j]-a[i][j];
         }
     }
```

NAME-SHREYAS NEHE
REG.NO-21BCE3359

```
int sequence_whichissafeUENCE()
{
    int i,j,k;
    int found=0;
    int count=0;
    while(count<n){
        int find=0;
       mark:for(i=0;i<n;i++)
        {
            if(!f[i]){
                int exe=1;
                for(j=0;j<m;j++){
                    if(needd[i][j]>w[j]){
                        exe=0;
                        break;
                    }
                }
                if(exe)
                {
                    for(k=0;k<m;k++)
                    {
                        w[k]+=a[i][k];
                    }
                    f[i]=1;
                    safseq[count]=i;
                    count++;
                    found=1;
                    goto mark;
                }
            }
        }
```

```
          if(!found){
              return 0;
          }
      }
      return 1;
  }


int st()
{
    init();
    if(sequence_whichissafeUENCE())
    {

        printf("safe\n");
        for(int i=0;i<n;i++){
        printf("%d",safseq[i]);
        printf(" ");
        }
        printf("\n");
        return 1;

    }
    else
    {
        printf("System is in unsafe state\n");
        return 0;
    }
}

int reqq(int process,int req[])
```

```
{
      int i;
      for(i=0;i<m;i++)
      {
          if(req[i]>needd[process][i])
          return 0;
          if(req[i]>whichisavailablee[i])
          return 0;

      }
      return 1;
}

void grant(int process,int req[])
{
    int i=0;
    for(i=0;i<m;i++)
    {
        whichisavailablee[i]-=req[i];
        a[process][i]+=req[i];
        needd[process][i]-=req[i];
    }
}

void print(int matrix[100][100])
{
    int i,j;
    for(i=0;i<n;i++)
    {
        for(j=0;j<m;j++){
            printf("%d",matrix[i][j]);
        }
```

NAME-SHREYAS NEHE
REG.NO-21BCE3359

```
            printf("\n");
        }
    }

    int main()
    {
     printf("Number of processes:")
    scanf("%d",&n);
   printf("Number of resource types:")
    scanf("%d",&m);
  printf("enter allocation matrix:")
    for(i=0;i<n;i++)
    {
        for(j=0;j<m;j++)
        scanf("%d",&a[i][j]);
    }
     printf("enter max matrix:")
    for(i=0;i<n;i++)
    {
        for(j=0;j<m;j++)
        scanf("%d",&highest[i][j]);
    }
     printf("enter available matrix:")
    for(i=0;i<m;i++)
     scanf("%d",&whichisavailablee[i]);
     int process;
     scanf("%d",&process);
     int reqqqq[100];
     for(i=0;i<m;i++)
     scanf("%d",&reqqqq[i]);

     int ISSAFE=st();
```

```
        printf("\n");
        if(ISSAFE && reqq(process,reqqqq)){

        grant(process,reqqqq);
        printf("safe\n");
        for(i=0;i<n;i++){
        printf("%d",safseq[i]);
        printf(" ");
        }
        printf("\n");
        printf("granted\n");
        }
        else
        {
            printf("not granted\n");
        }
        return 0;

}
```

OUTPUT WITH CODE:

NAME-SHREYAS NEHE
REG.NO-21BCE3359

```
#include<stdio.h>
#include<iostream>
using namespace std;
    int n,m;
    int i,j;
    int a[100][100];
    int highest[100][100];
    int whichisavailablee[100];
    int needd[100][100];
    int w[100];
    int f[100];
    int safseq[100];

    void init()
    {
        int i,j;
        for(i=0;i<m;i++)
        w[i]=whichisavailablee[i];

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

        for(i=0;i<n;i++)
        {
            for(j=0;j<m;j++)
            needd[i][j]=highest[i][j]-a[i][j];
        }
    }

    int sequence_whichissafeUENCE()
    {
        int i,j,k;
        int found=0;
        int count=0;
        while(count<n){
            int find=0;
           mark:for(i=0;i<n;i++)
            {
                if(!f[i]){
                    int exe=1;
                    for(j=0;j<m;j++){
                        if(needd[i][j]>w[j]){
                            exe=0;
                            break;
                        }
```

Ln 8, Col 32 Spaces: 4 UTF-8 CRLF {} C

NAME-SHREYAS NEHE
REG.NO-21BCE3359

```
                    }
                    if(exe)
                    {
                        for(k=0;k<m;k++)
                        {
                            w[k]+=a[i][k];
                        }
                        f[i]=1;
                        safseq[count]=i;
                        count++;
                        found=1;
                        goto mark;
                    }
                }
            }

            if(!found){
                return 0;
            }
        }
        return 1;
    }


int st()
{
    init();
    if(sequence_whichissafeUENCE())
    {

        printf("safe\n");
        for(int i=0;i<n;i++){
        printf("%d",safseq[i]);
        printf(" ");
        }
        printf("\n");
        return 1;

    }
    else
    {
        printf("System is in unsafe state\n");
        return 0;
    }
}
```

Ln 8, Col 32

NAME-SHREYAS NEHE
REG.NO-21BCE3359

```
}

int reqq(int process,int req[])
{
     int i;
     for(i=0;i<m;i++)
     {
         if(req[i]>needd[process][i])
         return 0;
         if(req[i]>whichisavailablee[i])
         return 0;

     }
     return 1;
}

void grant(int process,int req[])
{
    int i=0;
    for(i=0;i<m;i++)
    {
        whichisavailablee[i]-=req[i];
        a[process][i]+=req[i];
        needd[process][i]-=req[i];
    }
}

void print(int matrix[100][100])
{
    int i,j;
    for(i=0;i<n;i++)
    {
        for(j=0;j<m;j++){
            printf("%d",matrix[i][j]);
        }
        printf("\n");
    }
}

int main()
{
 printf("Number of processes:")
scanf("%d",&n);
printf("Number of resource types:")
```

NAME-SHREYAS NEHE
REG.NO-21BCE3359

input

```
Number of processes:5
Number of resource types:3
enter allocation matrix:0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
enter max matrix:7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
enter available matrix:3 3 2
1
1 0 2
safe
1 3 0 2 4

safe
1 3 0 2 4
granted


...Program finished with exit code 0
Press ENTER to exit console.
```

NAME-SHREYAS NEHE
REG.NO-21BCE3359

```
    printf("Number of resource types:");
     scanf("%d",&m);
   printf("enter allocation matrix:");
     for(i=0;i<n;i++)
     {
         for(j=0;j<m;j++)
         scanf("%d",&a[i][j]);
     }
      printf("enter max matrix:")
     for(i=0;i<n;i++)
     {
         for(j=0;j<m;j++)
         scanf("%d",&highest[i][j]);
     }
      printf("enter available matrix:")
     for(i=0;i<m;i++)
      scanf("%d",&whichisavailablee[i]);
      int process;
      scanf("%d",&process);
      int reqqqq[100];
      for(i=0;i<m;i++)
      scanf("%d",&reqqqq[i]);

      int ISSAFE=st();
      printf("\n");
      if(ISSAFE && reqq(process,reqqqq)){

      grant(process,reqqqq);
      printf("safe\n");
      for(i=0;i<n;i++){
      printf("%d",safseq[i]);
      printf(" ");
      }
      printf("\n");
      printf("granted\n");
      }
      else
      {
          printf("not granted\n");
      }
      return 0;

}
```