19/7/23

Write a C program to shedule Producer Consumer proce

```
#include <stdio.h>
#include <stdlib.h>
int mutex=1,full=0,empty=3,x=0;
void producer();
void consumer();
int wait(int);
int signal(int);

int main(){
 int n;
           n!=3
 while(1){
   printf("\n1.Producer\n2.Consumer\n3.Exit");
   printf("\n Enter Your choice ");
   scanf("%d",&n);
   switch(n){
  Case 1: if(mutex==1)&&empty(!=0))
            producer();
        else
          printf("Buffer is full!");
        break;
  Case 2: if(mutex==1)&&(full!=0)
          consumer();
        else
          printf("Buffer is empty!");
        break;
      }
   }
 return 0;
}
```

```
int wait(int s){
   return (--s);
  }
int signal(int s) {
    return (++s)
  }
void producer() {
  mutex = wait(mutex);
   full=signal (full);
   empty = wait(empty);
    x++;
   printf("\n Producer produced the item %d", x);
   mutex = signal(mutex);
 }
void consume () {
  mutex = wait (mutex);
  full = wait (full);
  empty = signal (empty);
  printf("\n Consumer consumes item %d", x);
   x--;
  mutex = signal (mutex);
}
```

Output:-

```
1. Producer 2. Consumer 3. Exit
Enter your choice : 1
 Producer produces item 1
Enter your choice : 2
 Consumer Consume item 1
Enter your choice : 2
Buffer is empty ! !
Enter your choice : 3
  Exit
```

19/7/23

Write a C prog

```
#include<stdio.h>
#include <stdlib.h
int at[10], i, n, d

void edf(){
int time = lcm A
int op = 0, pr = 0
int flag, i;
while (op< = tim
    for (i = 0; i<
        if (
    }
   }
  flag = 0;
 for (i=0; i<n
     if (
     flag
   }}   break
 if (flag =
    pr =
 } else {

    for

            }
```

7/23

Write a C program to simulate Banker algorithm to prevent deadlock.

```
#include<stdio.h>
#include<stdlib.h>
int main(){
int n,m,allo[10][10],req[10][10],ava[10],need[10][10],i,j,k,
    flag[10],prev[10],c,count=0,z=0,arr[10];

printf("Enter the no of processes & no of resources required\n");
scanf("%d %d",&n,&m);
printf("Enter total no of required resources %d for each process\n",n);
for(i=0;i<n;i++)
  for(j=0;j<m;j++)
    scanf("%d",&~~all~~req[i][j]);

printf("Enter the no of ~~process~~ allocated resources %d for each process\n",n);
  for(i=0;i<n;i++)
    for(j=0;j<m;j++)
      scanf("%d",&all[i][j]);

printf("Enter the no of available resources\n");
  for(i=0;i<m;i++)
     scanf("%d",&ava[i]);

  for(i=0;i<n;i++)
    for(j=0;j<m;j++)
      need[i][j]=req[i][j]-all[i][j];

  printf("\nNeed Matrix");
  for(i=0;i<n;i++){
     for(j=0;j<m;j++)
        printf("%d",need[i][j]);
     printf("\n");
  }
```

(3) 14/7/23

```
for(i=0; i<n; i++)
    flag[i] = 1;

R= 1;

while(R){
 R=0;
 for(i=0; i<n; i++){
   if (flag[i]){
       c=0;
     for(j=0;j<m;j++){
      if(needed[i][j] <= ava[j]){
          c++;
      } }

   if(c==m){
    arr[z++] =i;
   printf("Resources can be allocated to Process: %d &
   available resources are  ",(i+1));
   for(j=0;j<n; j++){
       printf("%d", ava[j]);
   } printf("\n");
   for(j=0; j<m ; j++){
       ava[j] += all[i][j];
       all[i][j]=0;
   }
   flag[i]=0;
   count++;
   }
 }}

 for(i=0;i<n;i++){
   if (flag[i]! = prev[i]){
       R= 1;
       break;
 } }
}
for(i=0; i<n ;i++)
      prev [i] = flag[i
printf("<");
   for(i=0;i<n; i+
       printf(" %d
   printf(">");

 if (count ==n){
      printf("\n Sys
 }else{
     printf(" \n Sys
 }
 return0;
}
```

Output:

```
Enter no of process & no
 5 3
Enter total no of reso
  7 5 3
  3 2 2
  9 0 2
  2 2 2
  4 3 3
Enter no of allocated
  0 1 0
  2 0 0
  3 0 2
  2 1 1
  0 0 2
Need matrix
  7 4 3
  1 2 2
  6 0 0
  0 1 1
  4 3 1
Resources can be alloca
Resources can be allocat
Resources can be alloca
Resources can be alloca
Resources can be allo
Safe Sequence <2 4 5
System is in safe mo
```

```
for(i=0; i<n ;i++)
    ans[i] = flag[i];

printf("<");
  for(i=0; i<n; i++)
      printf("%d", ans[i]);
printf(">");

if (count ==n){
      printf("\n System is in safe mode \n No Deadlock");
}else{
      printf("\n System is not in safe mode Deadlock found
}
return0;
}
```

output:

```
Enter no of process & no of resource required
 5 3
Enter total no of required resours for each proc
 7 5 3
 3 2 2
 9 0 2
 2 2 2
 4 3 3
Enter no of allocated resource for each proc
 0 1 0
 2 0 0
 3 0 2
 2 1 1
 0 0 2
Need matrix
 7 4 3
 1 2 2
 6 0 0
 0 1 1
 4 3 1
Resources can be allocated to P2 & available are : 3 3 2
Resources can be allocated to P4 : & available are : 5 3 2
Resources can be allocated to P5 & available are 7 4 3
Resources can be allocated to P1 & available are 7 4 5
Resources can be allocated to P3 & available are 7 5 5
Resource can be allocated
Safe sequence <2 4 5 1 3>
System is in safe mode No Deadlock
```

Write a C program to Simulate Dining Philosopher

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

#define N 5
# define THINKING 2
# define HUNGRY 1
#define EATING 0
# define LEFT (phnum +4) % N
# define RIGHT (phnum +1) % N

int state[N];
int phil[N]= {0,1,2,3,4};

sem_t mutex;
sem_t S[N];

void test (int phnum) {
  if (state[phnum] == HUNGRY && state[LEFT] != EATING
    && state [RIGHT] != EATING){
      state (phnum] = EATING;

    sleep(2);
   printf(" Philosopher %d takes fork %d & %d \n",
              phnum +1, LEFT +1, phnum +1);
  printf (" philosopher %d is Eating \n", phnum +1);
  sem_post (& S[phnum]);
  }
}
void take_fork(int phnum){
    sem_wait (& mutex);
    state [phnum] = HUNGRY;
    printf (" Philosopher %d is Hungry \n," phnum +1);
```

```
test (phnum);
sem_post (& mutex);
sem_wait(& S[phnum]
  sleep(1);
}
void put_fork (int ph
   sem_wait (& mute
   state (phnum] = THI
  printf (" Philosoph
         phnum +1, L
 printf (" Philosoph

  test ([LEFT);
  test(RIGHT);
  sem_post(& mute
}
void *philosoph
   while(1) {
     int * i =
      sleep (1
      take_for
      sleep (0
     put_fork
   }
 }

int main(){
  int i;
  pthread_t th
  sem_init (
   test (phnum);
  sem_post (&mutex);
  sem_wait(&S[phnum]);
   sleep(1);
  }
void put_fork (int phnum) {
    sem_wait (& mutex);
   state [phnum] = THINKING;
  printf ("Philosopher %d putting fork %d & %d down\n",
          phnum +1, LEFT +1, phnum + 1);
 printf ("Philosopher %d is thinking\n", phnum + 1);

  test ([LEFT]);
  test (RIGHT);
  sem_post(&mutex);
 }
void *philospher (void * num) {
    while (1) {
        int * i = num;
         sleep (1);
        take_fork (* i);
        sleep (0);
       put_fork (* i);
    }
 }
int main () {
  int i;
  pthread_t thread_id [N];
  sem_init (& mutex, 0, 1);
```

```
for (i=0; i<N; i++)
    sem_init (&S[i],0,0);
for (i=0; i<N ;i++){
    pthread-creat (&thread_id[i], NULL, philosopher, &phil[i]);
  printf ("philosopher %d is thinking\n", i+1);
}
for (i=0;i<N; i++)
    pthread-join (thread-id[i], NULL);
}
```

output

```
Philosopher 4 takes fork 3 & 4
Philosopher 4 is Eating
Philosopher 3 is hungry
Phylosopher 2 is putting fork 1 & 2 down
Philosopher 2 is thinking
Philosopher 1 takes for 5 & 1
Philosopher 1 is Eating
Philosopher 3 is hungry
Philosopher 4 is putting fork 3 & 4 down
Philosopher 4 is thinking
Philosopher 3 takes fork 2 & 3 _
```

31/7/23

Write a C progrm

```
#include <stdio.h
void main( )
{ int n,m,allo[
  i,j,k, flag[10]

printf("Enter the
  scanf("%d %d
printf("Enter the
  for(i=0; i<n
    for(j=0; j<
      scanf("%
printf(" Enter a
for(i=0; i<n;
  for(j=0;j<n
    scanf("%
printf("Enter
  for (i=0;
      sc

  for(i=0;
    for(j=0;
      need[

  k=1;
while (k) {
k=0;
for (i=0;i<n
  if(flag[i
    (
```