WEEK 5

- a)Write a C program to simulate the concept of Dining-Philosophers problem.
- b)Write a C program to simulate producer-consumer problem using semaphores.

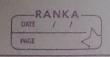
```
a)CODE:
#include <pthread.h>
#include <semaphore.h>
#include <stdio.h>
#define N 5
#define THINKING 2
#define HUNGRY 1
#define EATING 0
#define LEFT (phnum + 4) % N
#define RIGHT (phnum + 1) % N
int state[N];
int phil[N] = \{0, 1, 2, 3, 4\};
sem t mutex;
sem_t S[N];
void test(int phnum)
      if (state[phnum] == HUNGRY
             && state[LEFT] != EATING
             && state[RIGHT] != EATING) {
             state[phnum] = EATING;
            sleep(2);
             printf("Philosopher %d takes fork %d and %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 phnum)
      sem_wait(&mutex);
      state[phnum] = THINKING;
      printf("Philosopher %d putting fork %d and %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* philosopher(void* num)
```

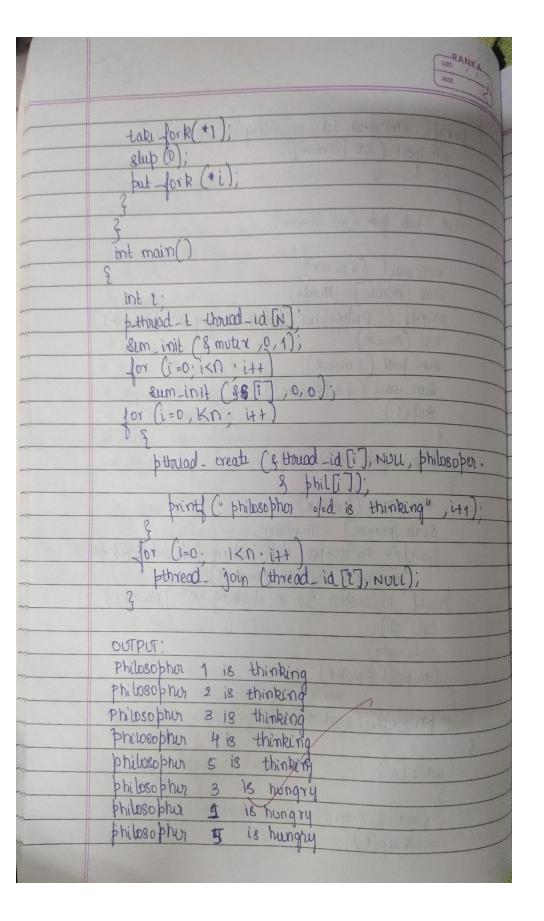
```
{
       while (1) {
              int* i = num;
              sleep(1);
              take_fork(*i);
              sleep(0);
              put_fork(*i);
       }
}
int main()
       int i;
       pthread_t thread_id[N];
       sem_init(&mutex, 0, 1);
       for (i = 0; i < N; i++)
              sem_init(&S[i], 0, 0);
       for (i = 0; i < N; i++) {
              // create philosopher processes
              pthread_create(&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);
}
```

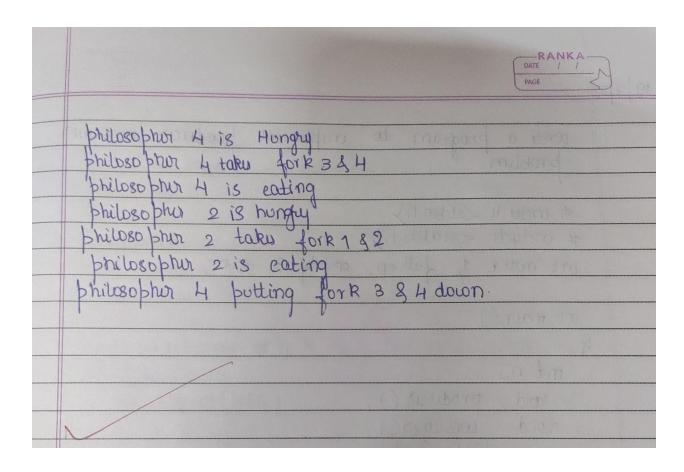
OBSERVATION:

19 7 23		implement Dinning-philosol
781	Write a program to	A CONTRACTOR OF THE PARTY OF TH
731	Write a program to	A CONTRACTOR OF THE PARTY OF TH
- Tal (Write a program to	imbliment Dinning - bhill-
3	Write a program to	implement Dinning - hhile
781 1.	problem.	7 (1080)
		CAMPAGE CONTRACT
		01301
		a faaggege as
	# include < semophore.h)	7 02 Ubi \$1 09
	# include <8tdio.h>	
	h : =	that John choise:
	# dyine N5	1. Monstonic
	# outine Thinking 2	103 s
	# define Hungry 1	B. Excl
	# dyine cating o	
	# dyline left (Phnum+	4)% N M M M M M M M M M M M M M M M M M M
	# out ne Right (Phnum +	1)% No miliones who
	0.75	56 95
	int state [N];	ents diadune
9 18 P	int phil[N] = {0,1,2,3,	43;
1 011 1	sein_timuta;	08 18 18 18 18 18 18 18 18 18 18 18 18 18
	Sem_t s[N];	SES FA 994 HELDHUR
	void tet (int phnom)	
		(110)
	State [Phnum] == Hur State [left] ! State Righ]	ngry
	38 state [left]!	= leating
	State Right	= Eating
	State [Phnum] = Eati	ng.
	8 lup (2);	
	print (" philosophia ?	d take fork of d and old in"
	•	shown+1, yf+1 phown+1)



```
print (" Philosophus "I'd is Eating In" . phnum+1);
  sem-post (&S [Phnum]
void tak fork (int brown
   sem wait (& mutux):
   State [Phoum] = Hungly;
   print! (" Philosopha "lod is Hungry In" phnum+ 1
   tast (Phnum);
   Sum-post (& mutix);
    Sun-wait (& s [Phnum]
vaid but-fork(int phnum)
sem-wait (& motor)
    State [shown] = Thinking
    print! ( Philosopha old putting for k old & old down"
     phnum+1 Wt+1 phnum+1
   Print Philosophun old is thinking In , phnum+1
    tut (Wt):
    test (right).
   Sim post (f motiz); and and the
void* prilosophur (void * num)
   while (1
    buf & T= un,
     8hp/1
```





OUTPUT:

```
| Second | S
```

```
b)CODE:
#include<stdio.h>
#include<stdlib.h>
int mutex=1,full=0,empty=3,x=0;
int main()
  int n;
  void producer();
  void consumer();
  int wait(int);
  int signal(int);
  printf("\n1.Producer\n2.Consumer\n3.Exit");
  while(1)
     printf("\nEnter 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;
       case 3:
             exit(0);
             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);
  printf("\nProducer produces the item %d",x);
  mutex=signal(mutex);
}
void consumer()
  mutex=wait(mutex);
  full=wait(full);
  empty=signal(empty);
  printf("\nConsumer consumes item %d",x);
  X--;
  mutex=signal(mutex);
}
```

OBSERVATION:

```
19 7 23
                 program to implement
                                             producir - consumer
         write a
          broblem
        # include <stdio.h}
        # include <stdlib.h}
        int mutic= 1, full=0, empty=3, x=0;
        int main ()
          int n;
            roid products ();
                  consumur ();
           roid
            int wait (int);
           int signal (int);
             print ( 1. producer \n 2. consumer \n 3. Exit );
             while
              print ("Enter your choice"
               scanf ("olod"
                Switch (n
                            (motin == ) 28 (empty!=0)
              case 1:
                           produces ?
                        else'
                        print (" Buffer is full"
               case 2 !
                         4 ((mutix==1) 88 (full 6=0)
                          " workumen (
                         else
                            printf
                                    " Buffer is empty")
                           break.
```

```
case 3: exit(0);
         break;
 return 0;
 int wait (ints)
  return (--8),
  int signal (ints)
  return (++8
 void products
    mutix = wait (motix);
      full = signal (full)
     empty = wait Cempty
      nett:
     print( " producer froduces the item % d", x);
     mutix = signal (mutix)
  void wasumer()
  mute x = wait (mutex);
    full = wait (full)
   empty - signal (empty);
   printf(" consumer consume item /d", x)
    mutin= signal (mutin);
```

	OVE ANKA PAGE
	The state of the s
	output!
	1. Producti
	2. Consumer
	3. Erit
	Entir your choice: 1
	producer produces the item 1
	(9 tai) 4,600 ta
	Enter your choice: 2
	Consumer consumes the item 2
	Entre unua choiu: 2
	Enter your Choice: 2 Buffer is empty.
	Buffer 18 empty.
	1 / / Marine and the first of the second sec
	V NOQ 1 1 2
	S 24/7/23
	7//
	Marie Marie
	11 HX
	Ly That metrical contract connection trust
	(stim) teach and a
	(A) Gringing that
	(etim\) seas y there
	The state of the s
	A VIII on Many a stay of the
-1-	The and mill make the second of the second

OUTPUT:

```
C:\Users\Admin\Desktop\bm21cs065\procons\bin\Debug\procons.exe

1.Producer
2.Consumer
3.Exit
Enter your choice:1

Producer produces the item 1
Enter your choice:2

Consumer consumes item 1
Enter your choice:2

Buffer is empty!!
Enter your choice:_
```