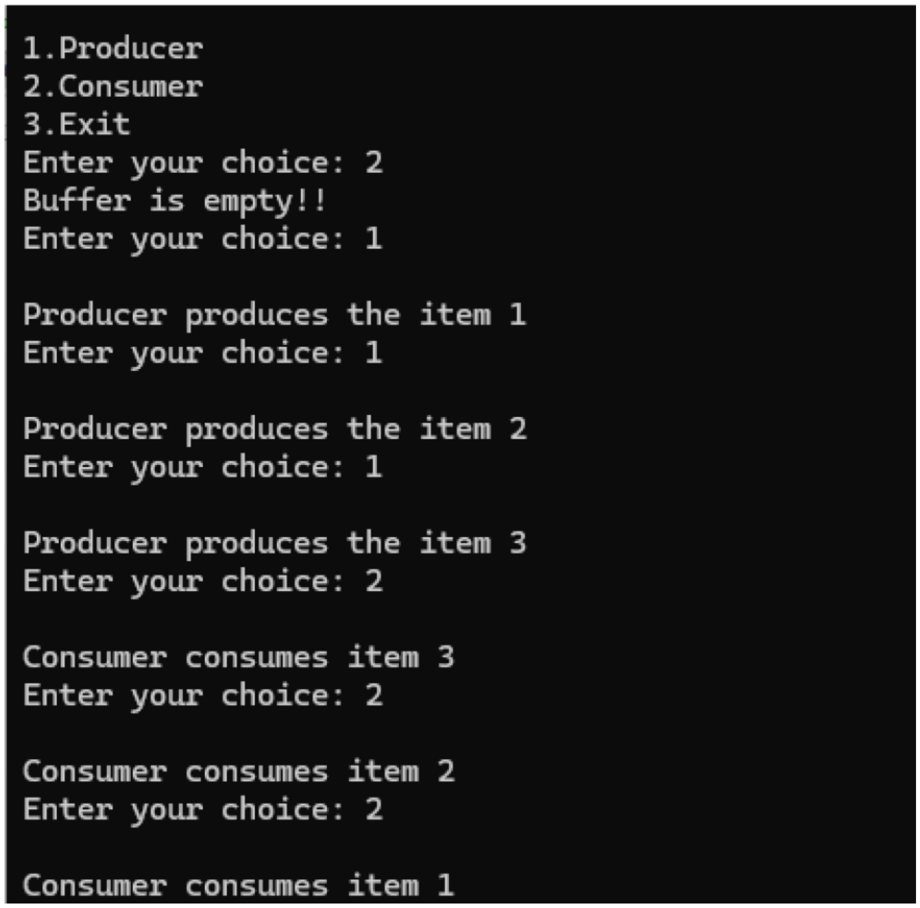
Program 4
Write a C program to simulate: a) producer consumer problem using semaphores   b) dining philosophers problem
Code:

```
#include<stdio.h>
#include<stdlib.h
>
int
mutex=1,full=0,e
mpty=3,x; int
main()
{
int n;
void
produ
cer();
void
consu
mer();
int
wait(i
nt);
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;
}
in
t
w
ai
t(i
nt
s)
{
re
tu
rn
(-
-
s)
;
}
```

```
in
t
si
g
na
l(i
nt
s)
{
r
e
t
u
r
n
(
+
+
s
);
}
void producer()
{
    mutex=wait(mutex);
    full=signal(full);
    empty=wait(empty);
    x++;
  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);
}
```

Output:



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

Producer produces the item 1
Enter your choice: 1

Producer produces the item 2
Enter your choice: 1

Producer produces the item 3
Enter your choice: 2

Consumer consumes item 3
Enter your choice: 2

Consumer consumes item 2
Enter your choice: 2

Consumer consumes item 1
```

b) Dining philosopher

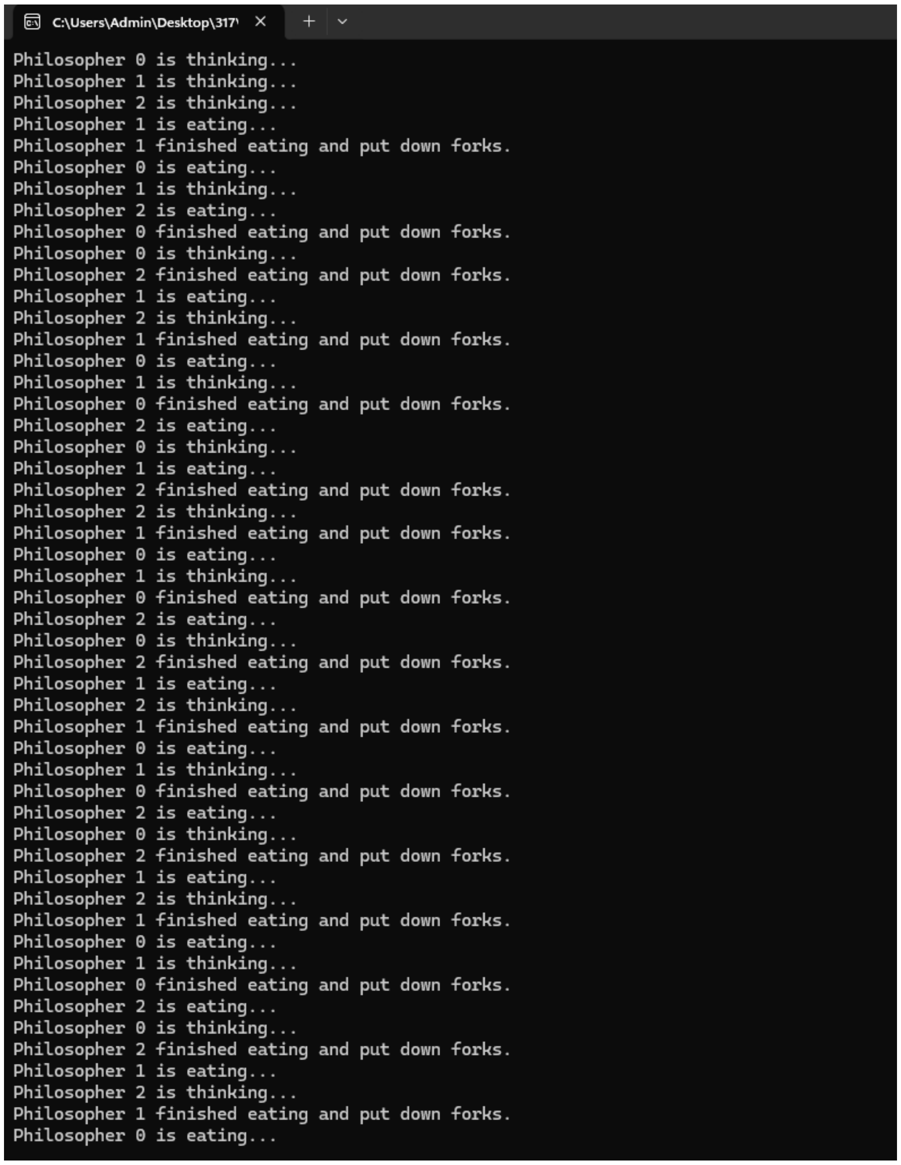
```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>
#define N 3
sem_t forks[N];
sem_t mutex;
void *philosopher(void *num) {
    int id = *(int *)num;

    while (1) {
        printf("Philosopher %d is thinking...\n", id);
        sleep(1);
        sem_wait(&mutex);
        sem_wait(&forks[id]);
        sem_wait(&forks[(id + 1) % N]);
        sem_post(&mutex);
        printf("Philosopher %d is eating...\n", id);
        sleep(2);
        sem_post(&forks[id]);
        sem_post(&forks[(id + 1) % N]);
        printf("Philosopher %d finished eating and put down forks.\n", id);
        sleep(1);
    }
}

int main() {
    pthread_t tid[N];
    int ids[N];
    for (int i = 0; i < N; i++) {
        sem_init(&forks[i], 0, 1);
    }
    sem_init(&mutex, 0, 1);
    for (int i = 0; i < N; i++) {
        ids[i] = i;
        pthread_create(&tid[i], NULL, philosopher, &ids[i]);
    }

    for (int i = 0; i < N; i++) {
        pthread_join(tid[i], NULL);
    }
    for (int i = 0; i < N; i++) {
        sem_destroy(&forks[i]);
    }
    sem_destroy(&mutex);
    return 0;
}
```

Output:



```
Philosopher 0 is thinking...
Philosopher 1 is thinking...
Philosopher 2 is thinking...
Philosopher 1 is eating...
Philosopher 1 finished eating and put down forks.
Philosopher 0 is eating...
Philosopher 1 is thinking...
Philosopher 2 is eating...
Philosopher 0 finished eating and put down forks.
Philosopher 0 is thinking...
Philosopher 2 finished eating and put down forks.
Philosopher 1 is eating...
Philosopher 2 is thinking...
Philosopher 1 finished eating and put down forks.
Philosopher 0 is eating...
Philosopher 1 is thinking...
Philosopher 0 finished eating and put down forks.
Philosopher 2 is eating...
Philosopher 0 is thinking...
Philosopher 1 is eating...
Philosopher 2 finished eating and put down forks.
Philosopher 2 is thinking...
Philosopher 1 finished eating and put down forks.
Philosopher 0 is eating...
Philosopher 1 is thinking...
Philosopher 0 finished eating and put down forks.
Philosopher 2 is eating...
Philosopher 0 is thinking...
Philosopher 2 finished eating and put down forks.
Philosopher 1 is eating...
Philosopher 2 is thinking...
Philosopher 1 finished eating and put down forks.
Philosopher 0 is eating...
Philosopher 1 is thinking...
Philosopher 0 finished eating and put down forks.
Philosopher 2 is eating...
Philosopher 0 is thinking...
Philosopher 2 finished eating and put down forks.
Philosopher 1 is eating...
Philosopher 2 is thinking...
Philosopher 1 finished eating and put down forks.
Philosopher 0 is eating...
Philosopher 1 is thinking...
Philosopher 0 finished eating and put down forks.
Philosopher 2 is eating...
Philosopher 0 is thinking...
Philosopher 2 finished eating and put down forks.
Philosopher 1 is eating...
Philosopher 2 is thinking...
Philosopher 1 finished eating and put down forks.
Philosopher 0 is eating...
```