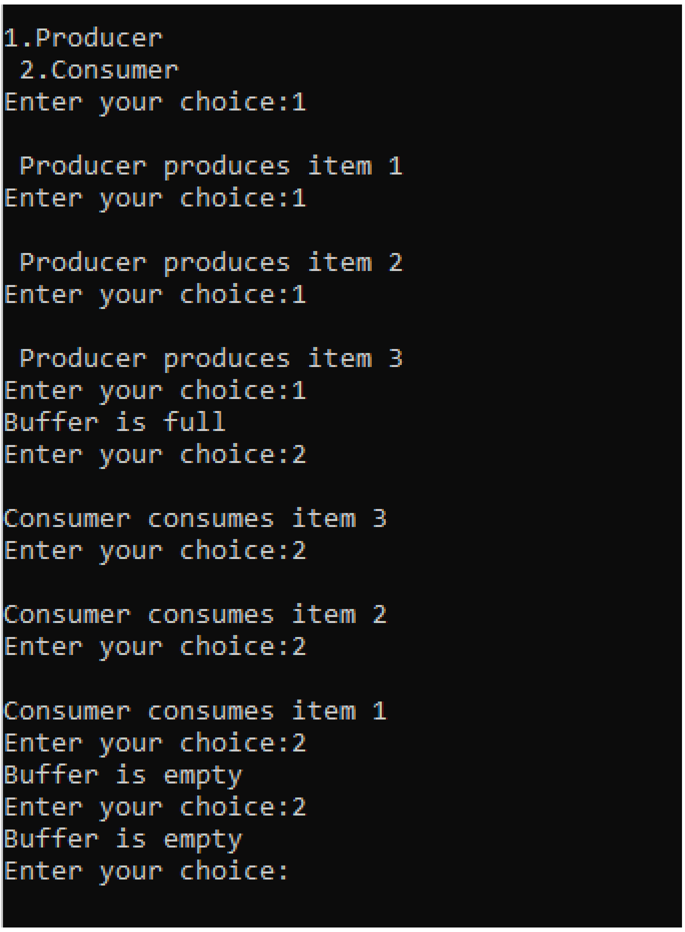
# PRODUCER-CONSUMER PROBLEM

**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 \n 2.Consumer");

    while(1)
    {
      printf("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;
          case 3:exit(0);break;


        }
      }
      return 0;
}//main
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 produces item %d",x);
      mutex=signal(mutex);
}
//producer
void consumer()
{
      mutex=wait(mutex);
      full=wait(full);
      empty=signal(empty);
      printf("\nConsumer consumes item %d",x);
      x--;
```

```
   mutex=signal(mutex);
}//consumer
```

**OUTPUT:**



```
1.Producer
 2.Consumer
Enter your choice:1

 Producer produces item 1
Enter your choice:1

 Producer produces item 2
Enter your choice:1

 Producer produces item 3
Enter your choice:1
Buffer is full
Enter your choice:2

Consumer consumes item 3
Enter your choice:2

Consumer consumes item 2
Enter your choice:2

Consumer consumes item 1
Enter your choice:2
Buffer is empty
Enter your choice:2
Buffer is empty
Enter your choice:
```

## 2.DINING PHILOSOPHERS:

**CODE:**

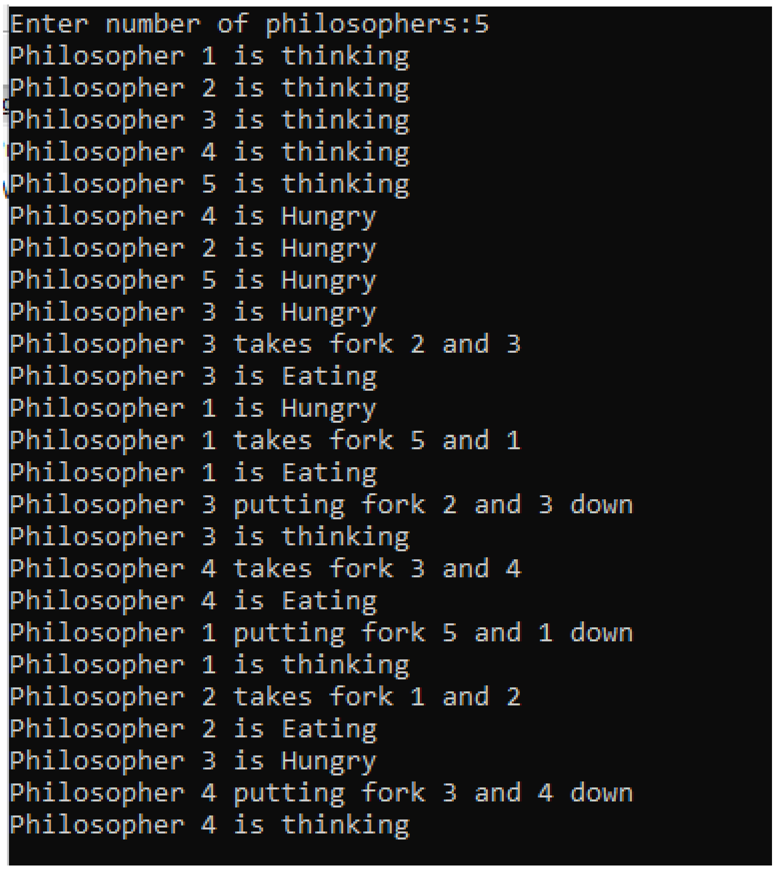
```
#include <pthread.h>

#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 N;
int state[10];
int phil[10] = { 0, 1, 2, 3, 4 };

sem_t mutex;
sem_t S[10];

void test(int phnum)
{
    if (state[phnum] == HUNGRY
        && state[LEFT] != EATING
        && state[RIGHT] != EATING) {
        // state that 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]) has no effect
        // during takefork
        // used to wake up hungry philosophers
        // during putfork
        sem_post(&S[phnum]);
    }
}

// take up chopsticks
```

```
void take_fork(int phnum)
{

        sem_wait(&mutex);

        // state that hungry
        state[phnum] = HUNGRY;

        printf("Philosopher %d is Hungry\n", phnum + 1);

        // eat if neighbours are not eating
        test(phnum);

        sem_post(&mutex);

        // if unable to eat wait to be signalled
        sem_wait(&S[phnum]);

        sleep(1);
}

// put down chopsticks
void put_fork(int phnum)
{

        sem_wait(&mutex);

        // state that thinking
        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()
{
printf("Enter number of philosophers:");
scanf("%d",&N);
```

```
        int i;
        pthread_t thread_id[N];

        // initialize the semaphores
        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);
}
```

**OUTPUT:**



```
Enter number of philosophers:5
Philosopher 1 is thinking
Philosopher 2 is thinking
Philosopher 3 is thinking
Philosopher 4 is thinking
Philosopher 5 is thinking
Philosopher 4 is Hungry
Philosopher 2 is Hungry
Philosopher 5 is Hungry
Philosopher 3 is Hungry
Philosopher 3 takes fork 2 and 3
Philosopher 3 is Eating
Philosopher 1 is Hungry
Philosopher 1 takes fork 5 and 1
Philosopher 1 is Eating
Philosopher 3 putting fork 2 and 3 down
Philosopher 3 is thinking
Philosopher 4 takes fork 3 and 4
Philosopher 4 is Eating
Philosopher 1 putting fork 5 and 1 down
Philosopher 1 is thinking
Philosopher 2 takes fork 1 and 2
Philosopher 2 is Eating
Philosopher 3 is Hungry
Philosopher 4 putting fork 3 and 4 down
Philosopher 4 is thinking
```