Product consumer

```
#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("\n 1. Producer\n 2. Consumer\n 3. Exit");
    while(1) {
    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;
    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);
        x++;
        printf ("\n Producer produces: %d", x);
        mutex = signal (mutex);
    }

void consumer() {
    mutex = wait(mutex);
    full = wait(full);
    empty = signal (empty);
    printf ("\n Consumer consums: %d", x);
    x--;
    mutex = signal (mutex);
}
```

OUTPUT

1. Producer
2. Consumer
3. Exit

Enter your choice: 1
Produces produces 1 item

Enter your choice: 2
consumer consumes 1 item

Enter your choice: 2
Buffer is empty!

Dining Philosopher

```
#include <stdio.h>
#include <pthread.h>
#include <semaphore.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[right] != eating && state[left] != 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]);
  }
}
```

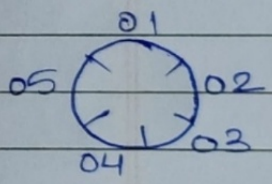
```
void put_fork (int phnum) {
    sem_wait (&mutex);
    state[phnum] = thinking;
printf ("philosopher %.d putting fork %.d
    & %.d d down \n", phnum+1, left+1,
                    phnum+1);
printf (" philosopher %.d is thinking\n",
                phnum+1);
test (left);
test (right);
 sem_post (&mutex);
}

void 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 philosopher (int *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++){
    pthread_create(&thread_id[i],
                   NULL, philosopher, &phi[i]);
    printf("philosopher %d is thinking\n",
                   i+1); }
    for(i=0; i<N; i++)
      pthread_join(thread_id[i], NULL);
      return 0;
}
```

OUTPUT-



philosopher 1 is thinking
2
3
4
5
philosopher 1 is hungry
3
2
philosopher 2 takes fork 1 and 2.
philosopher 2 starts eating
philospher 4 is hungry
philospher 5 is hungry
philospher 5 takes fork 4 and 5.
philospher 5 is eating.
philospher puts down fork 1 and 2.