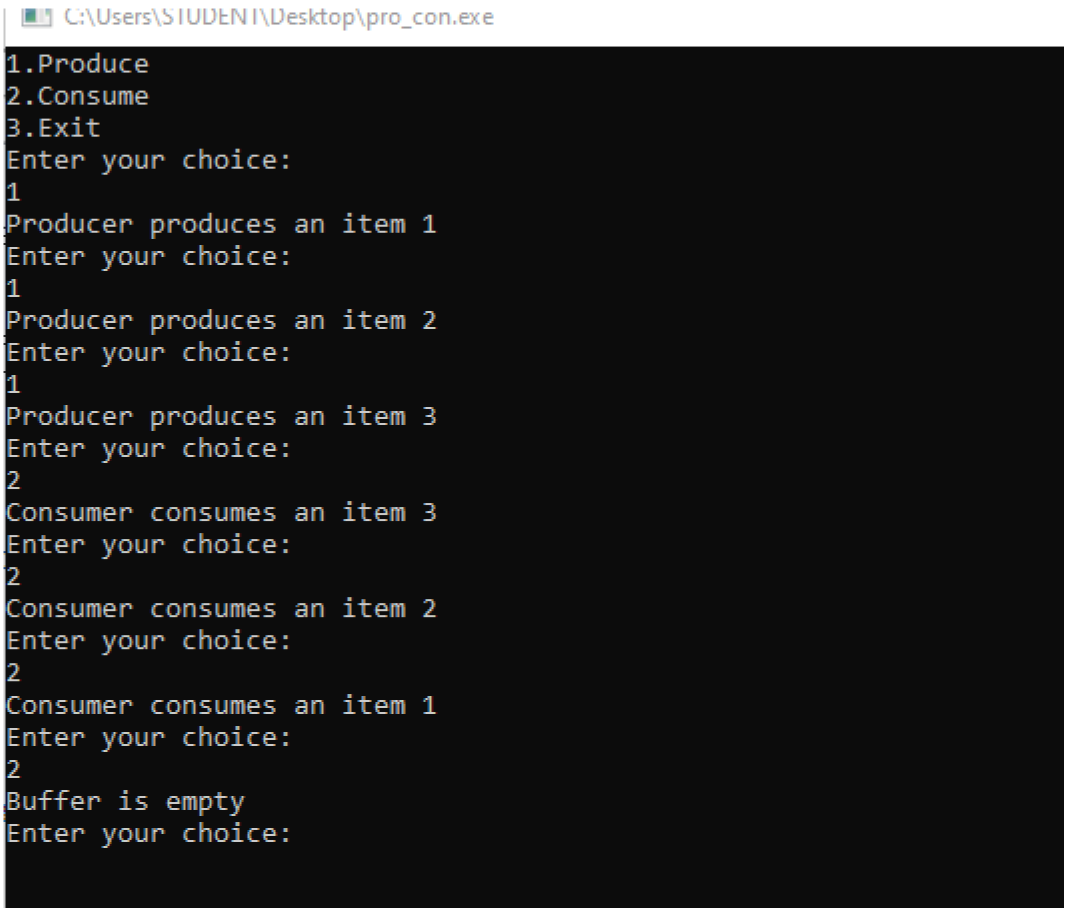
PRODUCER CONSUMER PROBLEM

```
#include<stdio.h>
#include<conio.h>
int mutex=1;
int full=0;
int empty=10;
int cnt=0;
int wait(int s)
{
while(s<=0);
s--;
return s;
}
int signal(int s)
{
s++;
return s;
}
void producer()
{
empty=wait(empty);
mutex=wait(mutex);
cnt++;
printf("Producer produces an item %d\n",cnt);
mutex=signal(mutex);
full=signal(full);
}
void consumer()
{
```

```
full=wait(full);
mutex=wait(mutex);
printf("Consumer consumes an item %d\n",cnt);
cnt--;
    mutex=signal(mutex);
empty=signal(empty);
}
void main()
{
int choice;
printf("1.Produce\n2.Consume\n3.Exit\n");
while(1)
{
printf("Enter your choice:\n");
scanf("%d",&choice);
switch(choice)
{
case 1:if(empty==0)
{
printf("Buffer is full\n");
}
else{
producer();
}
break;
case 2:if(full==0)
{
printf("Buffer is empty\n");
}
else{
consumer();
}
break;
case 3:exit(0);
     break;
default:printf("Invalid choice\n");
}
}
getch();


}
```



C:\Users\STUDENT\Desktop\pro_con.exe

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

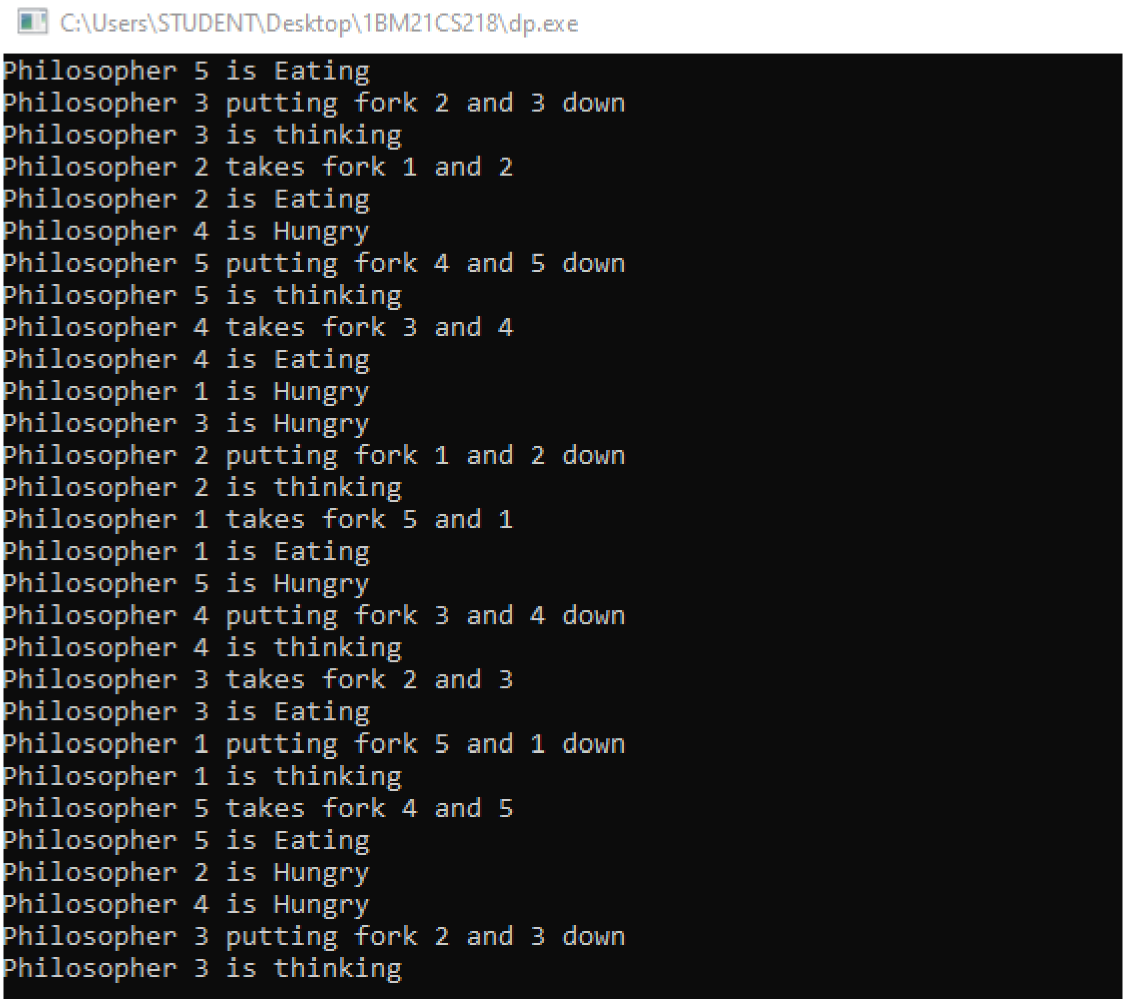
DINING PHILOSOPHER:

```
#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 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++) {
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);

}
```


C:\Users\STUDENT\Desktop\1BM21CS218\dp.exe

```
Philosopher 5 is Eating
Philosopher 3 putting fork 2 and 3 down
Philosopher 3 is thinking
Philosopher 2 takes fork 1 and 2
Philosopher 2 is Eating
Philosopher 4 is Hungry
Philosopher 5 putting fork 4 and 5 down
Philosopher 5 is thinking
Philosopher 4 takes fork 3 and 4
Philosopher 4 is Eating
Philosopher 1 is Hungry
Philosopher 3 is Hungry
Philosopher 2 putting fork 1 and 2 down
Philosopher 2 is thinking
Philosopher 1 takes fork 5 and 1
Philosopher 1 is Eating
Philosopher 5 is Hungry
Philosopher 4 putting fork 3 and 4 down
Philosopher 4 is thinking
Philosopher 3 takes fork 2 and 3
Philosopher 3 is Eating
Philosopher 1 putting fork 5 and 1 down
Philosopher 1 is thinking
Philosopher 5 takes fork 4 and 5
Philosopher 5 is Eating
Philosopher 2 is Hungry
Philosopher 4 is Hungry
Philosopher 3 putting fork 2 and 3 down
Philosopher 3 is thinking
```