VIT®

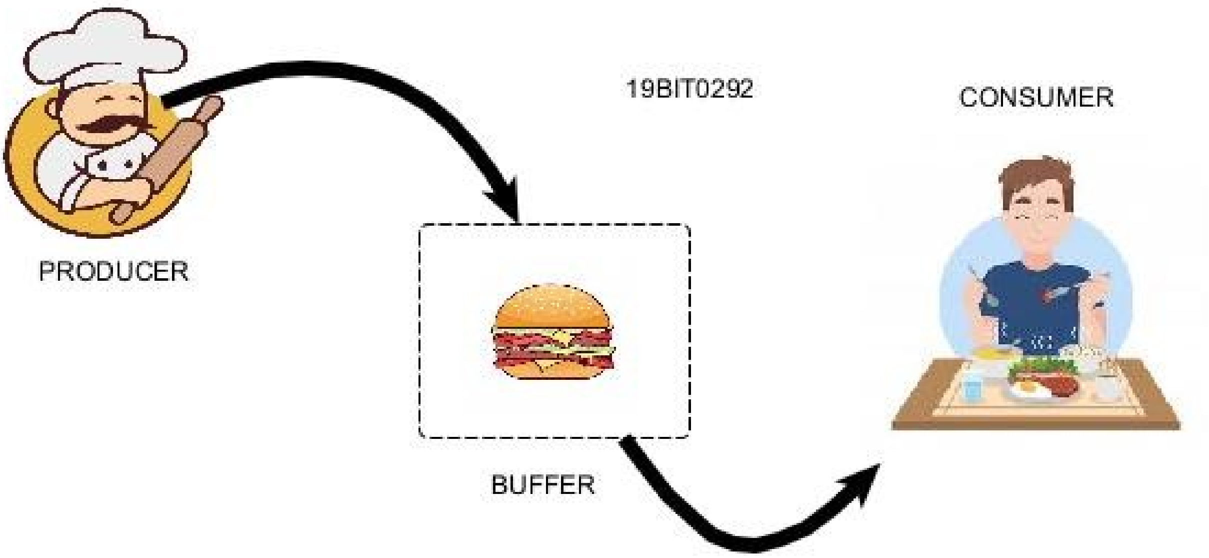
Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

19BIT0292
Bhaumik Tandan

ASSESMENT-2
OPERATING SYSTEM Laboratory

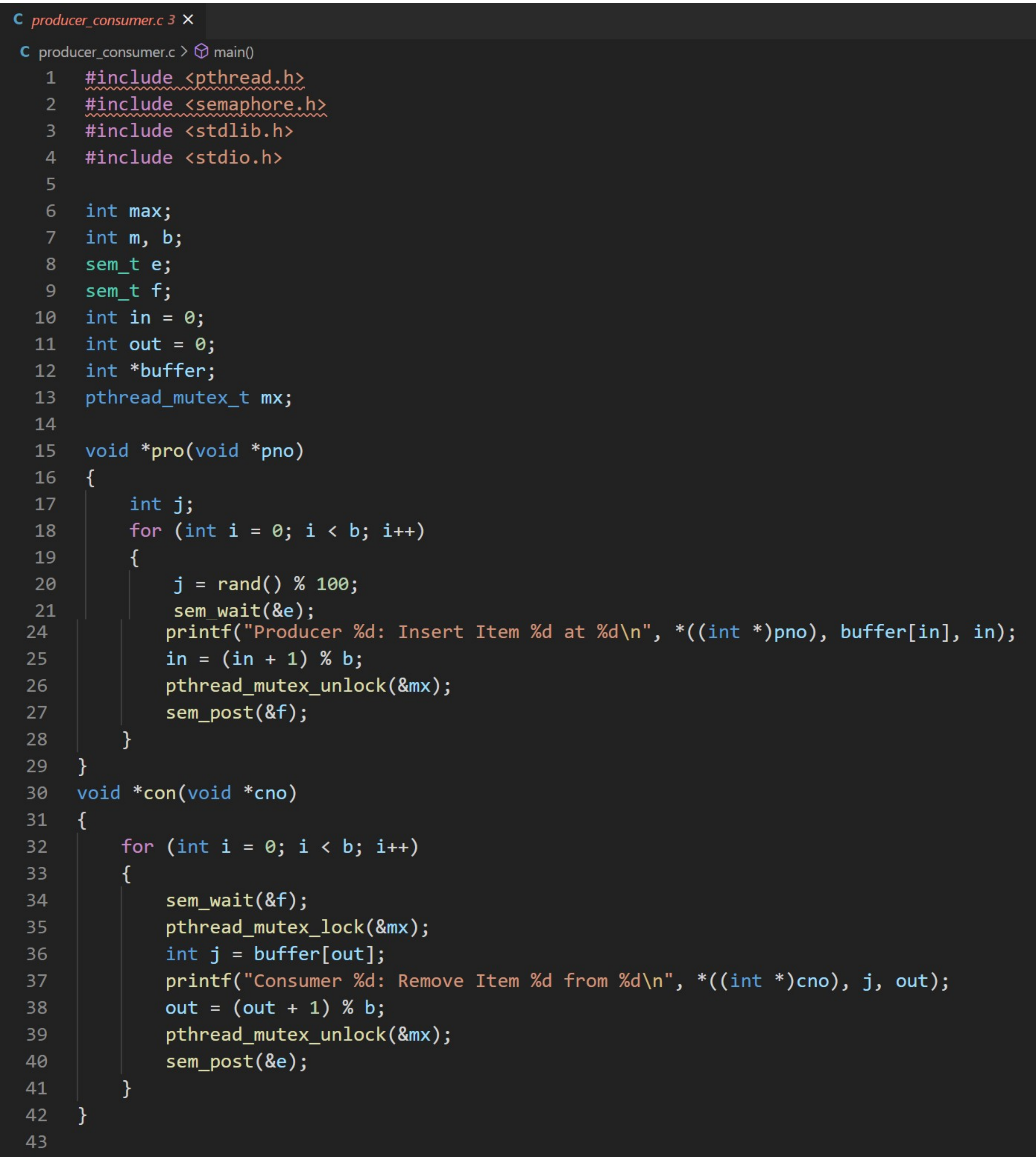
Q1. Write a program to implement the producer –consumer problem using semaphores.



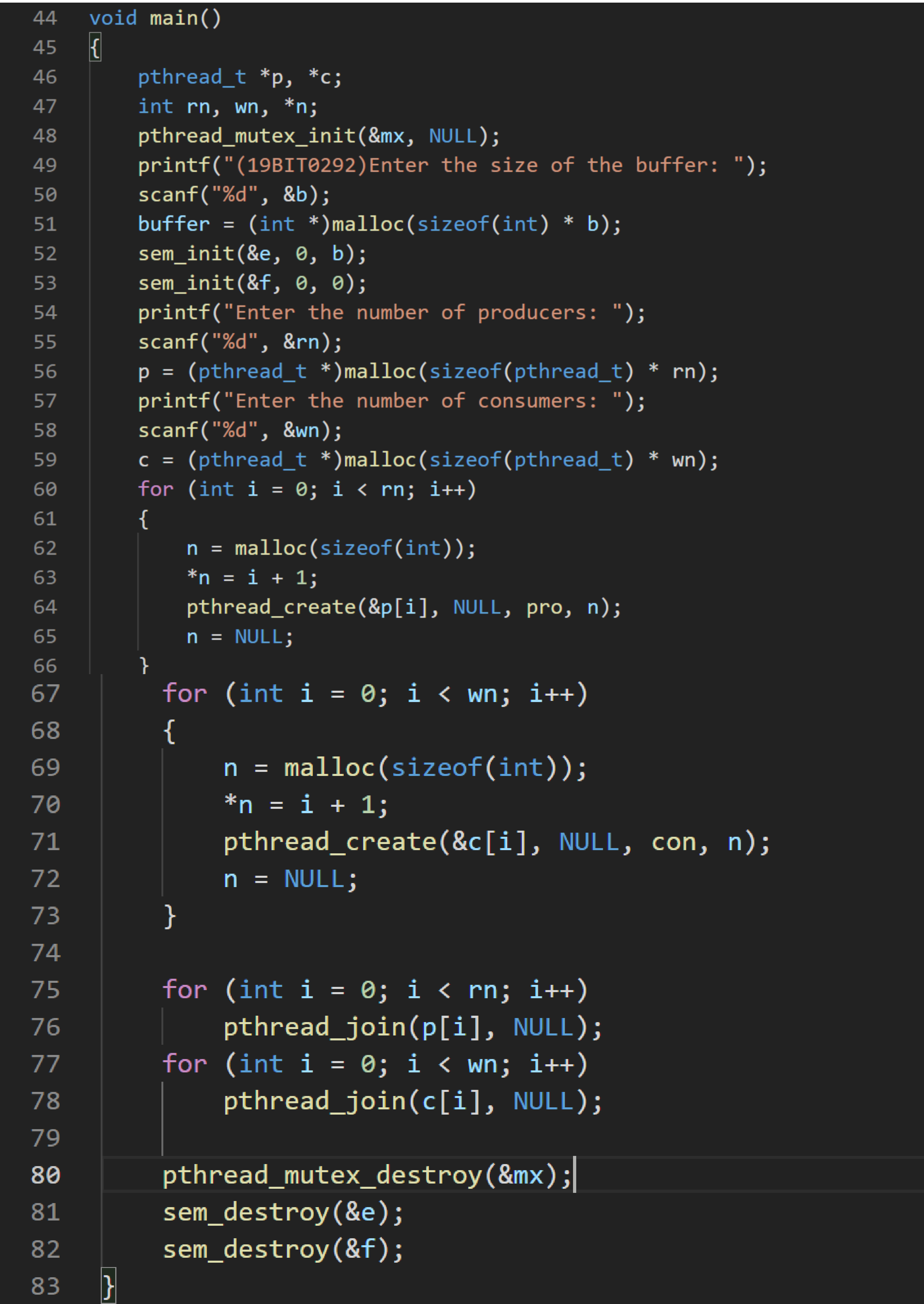
CODE

```
#include <pthread.h>
#include <semaphore.h>
#include <stdlib.h>
#include <stdio.h>

int max;
int m, b;
sem_t e;
sem_t f;
int in = 0;
int out = 0;
int *buffer;
pthread_mutex_t mx;

void *pro(void *pno)
{
    int j;
    for (int i = 0; i < b; i++)
    {
        j = rand() % 100;
        sem_wait(&e);
        pthread_mutex_lock(&mx);
        buffer[in] = j;
        printf("Producer %d: Insert Item %d at %d\n", *((int *)pno),
buffer[in], in);
        in = (in + 1) % b;
        pthread_mutex_unlock(&mx);
        sem_post(&f);
    }
}
void *con(void *cno)
{
    for (int i = 0; i < b; i++)
    {
        sem_wait(&f);
        pthread_mutex_lock(&mx);
        int j = buffer[out];
        printf("Consumer %d: Remove Item %d from %d\n", *((int *)cno),
j, out);
        out = (out + 1) % b;
        pthread_mutex_unlock(&mx);
        sem_post(&e);
    }
}

void main()
{
    pthread_t *p, *c;
    int rn, wn, *n;
    pthread_mutex_init(&mx, NULL);
    printf("(19BIT0292)Enter the size of the buffer: ");
    scanf("%d", &b);
    buffer = (int *)malloc(sizeof(int) * b);
    sem_init(&e, 0, b);
    sem_init(&f, 0, 0);
    printf("Enter the number of producers: ");
    scanf("%d", &rn);
    p = (pthread_t *)malloc(sizeof(pthread_t) * rn);
    printf("Enter the number of consumers: ");
    scanf("%d", &wn);
    c = (pthread_t *)malloc(sizeof(pthread_t) * wn);
    for (int i = 0; i < rn; i++)
    {
        n = malloc(sizeof(int));
        *n = i + 1;
        pthread_create(&p[i], NULL, pro, n);
        n = NULL;
    }
    for (int i = 0; i < wn; i++)
    {
        n = malloc(sizeof(int));
        *n = i + 1;
        pthread_create(&c[i], NULL, con, n);
        n = NULL;
    }

    for (int i = 0; i < rn; i++)
        pthread_join(p[i], NULL);
    for (int i = 0; i < wn; i++)
        pthread_join(c[i], NULL);

    pthread_mutex_destroy(&mx);
    sem_destroy(&e);
    sem_destroy(&f);
}
```

SCREENSHOT OF CODE

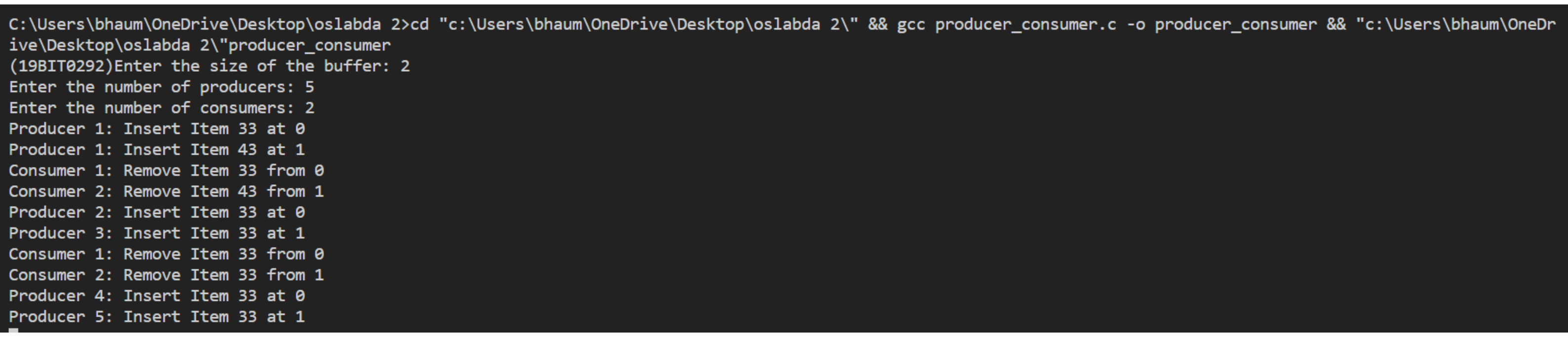


```c
#include <pthread.h>
#include <semaphore.h>
#include <stdlib.h>
#include <stdio.h>

int max;
int m, b;
sem_t e;
sem_t f;
int in = 0;
int out = 0;
int *buffer;
pthread_mutex_t mx;

void *pro(void *pno)
{
    int j;
    for (int i = 0; i < b; i++)
    {
        j = rand() % 100;
        sem_wait(&e);
        printf("Producer %d: Insert Item %d at %d\n", *((int *)pno), buffer[in], in);
        in = (in + 1) % b;
        pthread_mutex_unlock(&mx);
        sem_post(&f);
    }
}
void *con(void *cno)
{
    for (int i = 0; i < b; i++)
    {
        sem_wait(&f);
        pthread_mutex_lock(&mx);
        int j = buffer[out];
        printf("Consumer %d: Remove Item %d from %d\n", *((int *)cno), j, out);
        out = (out + 1) % b;
        pthread_mutex_unlock(&mx);
        sem_post(&e);
    }
}

```

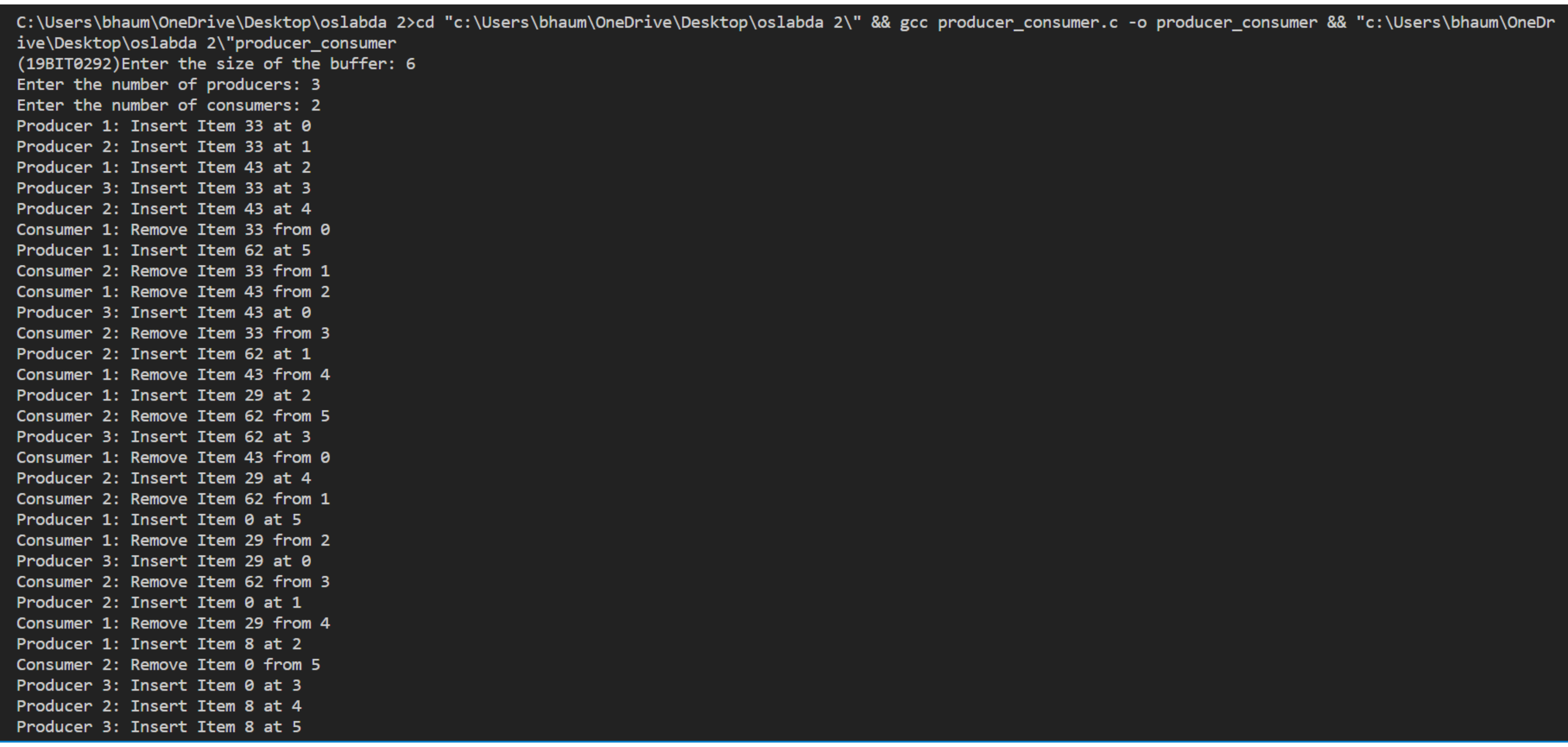


```c
void main()
{
    pthread_t *p, *c;
    int rn, wn, *n;
    pthread_mutex_init(&mx, NULL);
    printf("(19BIT0292)Enter the size of the buffer: ");
    scanf("%d", &b);
    buffer = (int *)malloc(sizeof(int) * b);
    sem_init(&e, 0, b);
    sem_init(&f, 0, 0);
    printf("Enter the number of producers: ");
    scanf("%d", &rn);
    p = (pthread_t *)malloc(sizeof(pthread_t) * rn);
    printf("Enter the number of consumers: ");
    scanf("%d", &wn);
    c = (pthread_t *)malloc(sizeof(pthread_t) * wn);
    for (int i = 0; i < rn; i++)
    {
        n = malloc(sizeof(int));
        *n = i + 1;
        pthread_create(&p[i], NULL, pro, n);
        n = NULL;
    }
    for (int i = 0; i < wn; i++)
    {
        n = malloc(sizeof(int));
        *n = i + 1;
        pthread_create(&c[i], NULL, con, n);
        n = NULL;
    }

    for (int i = 0; i < rn; i++)
        pthread_join(p[i], NULL);
    for (int i = 0; i < wn; i++)
        pthread_join(c[i], NULL);

    pthread_mutex_destroy(&mx);
    sem_destroy(&e);
    sem_destroy(&f);
}
```

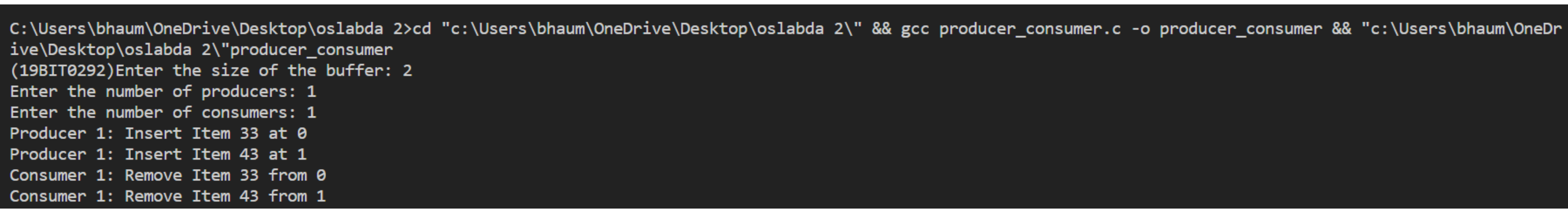
OUTPUT



```
C:\Users\bhaum\OneDrive\Desktop\oslabda 2>cd "c:\Users\bhaum\OneDrive\Desktop\oslabda 2\" && gcc producer_consumer.c -o producer_consumer && "c:\Users\bhaum\OneDr
ive\Desktop\oslabda 2\"producer_consumer
(19BIT0292)Enter the size of the buffer: 2
Enter the number of producers: 5
Enter the number of consumers: 2
Producer 1: Insert Item 33 at 0
Producer 1: Insert Item 43 at 1
Consumer 1: Remove Item 33 from 0
Consumer 2: Remove Item 43 from 1
Producer 2: Insert Item 33 at 0
Producer 3: Insert Item 33 at 1
Consumer 1: Remove Item 33 from 0
Consumer 2: Remove Item 33 from 1
Producer 4: Insert Item 33 at 0
Producer 5: Insert Item 33 at 1
```

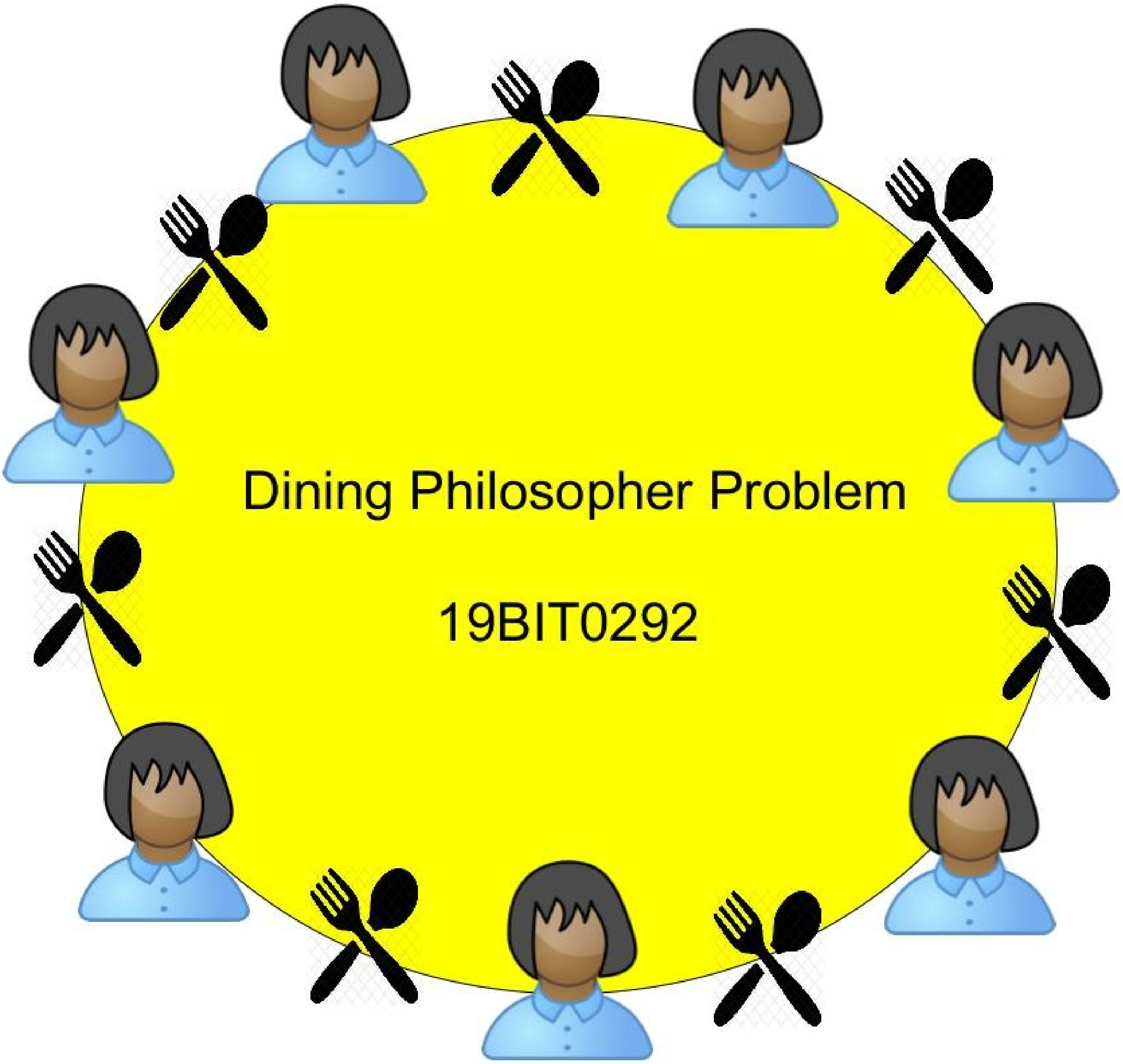


```
C:\Users\bhaum\OneDrive\Desktop\oslabda 2>cd "c:\Users\bhaum\OneDrive\Desktop\oslabda 2\" && gcc producer_consumer.c -o producer_consumer && "c:\Users\bhaum\OneDr
ive\Desktop\oslabda 2\"producer_consumer
(19BIT0292)Enter the size of the buffer: 6
Enter the number of producers: 3
Enter the number of consumers: 2
Producer 1: Insert Item 33 at 0
Producer 2: Insert Item 33 at 1
Producer 1: Insert Item 43 at 2
Producer 3: Insert Item 33 at 3
Producer 2: Insert Item 43 at 4
Consumer 1: Remove Item 33 from 0
Producer 1: Insert Item 62 at 5
Consumer 2: Remove Item 33 from 1
Consumer 1: Remove Item 43 from 2
Producer 3: Insert Item 43 at 0
Consumer 2: Remove Item 33 from 3
Producer 2: Insert Item 62 at 1
Consumer 1: Remove Item 43 from 4
Producer 1: Insert Item 29 at 2
Consumer 2: Remove Item 62 from 5
Producer 3: Insert Item 62 at 3
Consumer 1: Remove Item 43 from 0
Producer 2: Insert Item 29 at 4
Consumer 2: Remove Item 62 from 1
Producer 1: Insert Item 0 at 5
Consumer 1: Remove Item 29 from 2
Producer 3: Insert Item 29 at 0
Consumer 2: Remove Item 62 from 3
Producer 2: Insert Item 0 at 1
Consumer 1: Remove Item 29 from 4
Producer 1: Insert Item 8 at 2
Consumer 2: Remove Item 0 from 5
Producer 3: Insert Item 0 at 3
Producer 2: Insert Item 8 at 4
Producer 3: Insert Item 8 at 5
```



```
C:\Users\bhaum\OneDrive\Desktop\oslabda 2>cd "c:\Users\bhaum\OneDrive\Desktop\oslabda 2\" && gcc producer_consumer.c -o producer_consumer && "c:\Users\bhaum\OneDr
ive\Desktop\oslabda 2\"producer_consumer
(19BIT0292)Enter the size of the buffer: 2
Enter the number of producers: 1
Enter the number of consumers: 1
Producer 1: Insert Item 33 at 0
Producer 1: Insert Item 43 at 1
Consumer 1: Remove Item 33 from 0
Consumer 1: Remove Item 43 from 1
```

Q2. Write a Program to implement the solution for dining philosopher's problem.



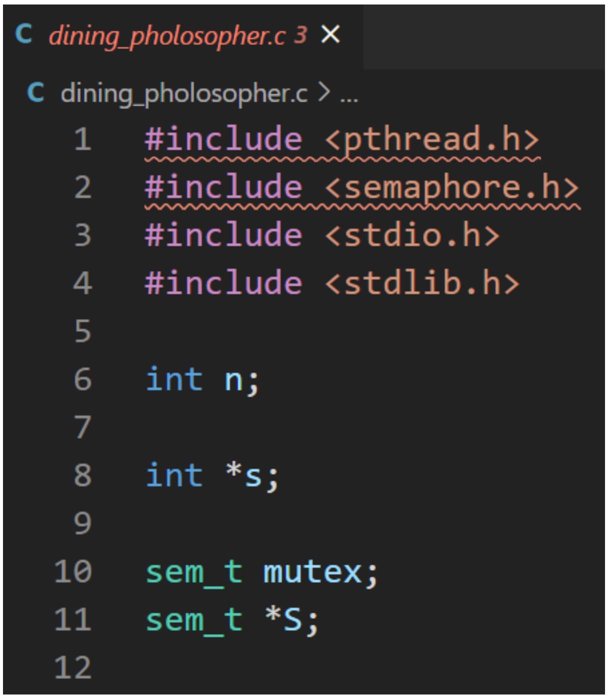
CODE

```
#include <pthread.h>

#include <semaphore.h>

#include <stdio.h>

#include <stdlib.h>


int n;


int *s;


sem_t mutex;

sem_t *S;


void test(int phnum)

{

    if (s[phnum] == 1 && s[(phnum + 4) % n] != 0 && s[(phnum + 1) % n] != 0)

    {

        s[phnum] = 0;


        printf("Philosopher %d takes fork %d and %d\n",

               phnum + 1, phnum + 1, (phnum + 4) % n + 1);


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


        sem_post(&S[phnum]);

    }

}
```

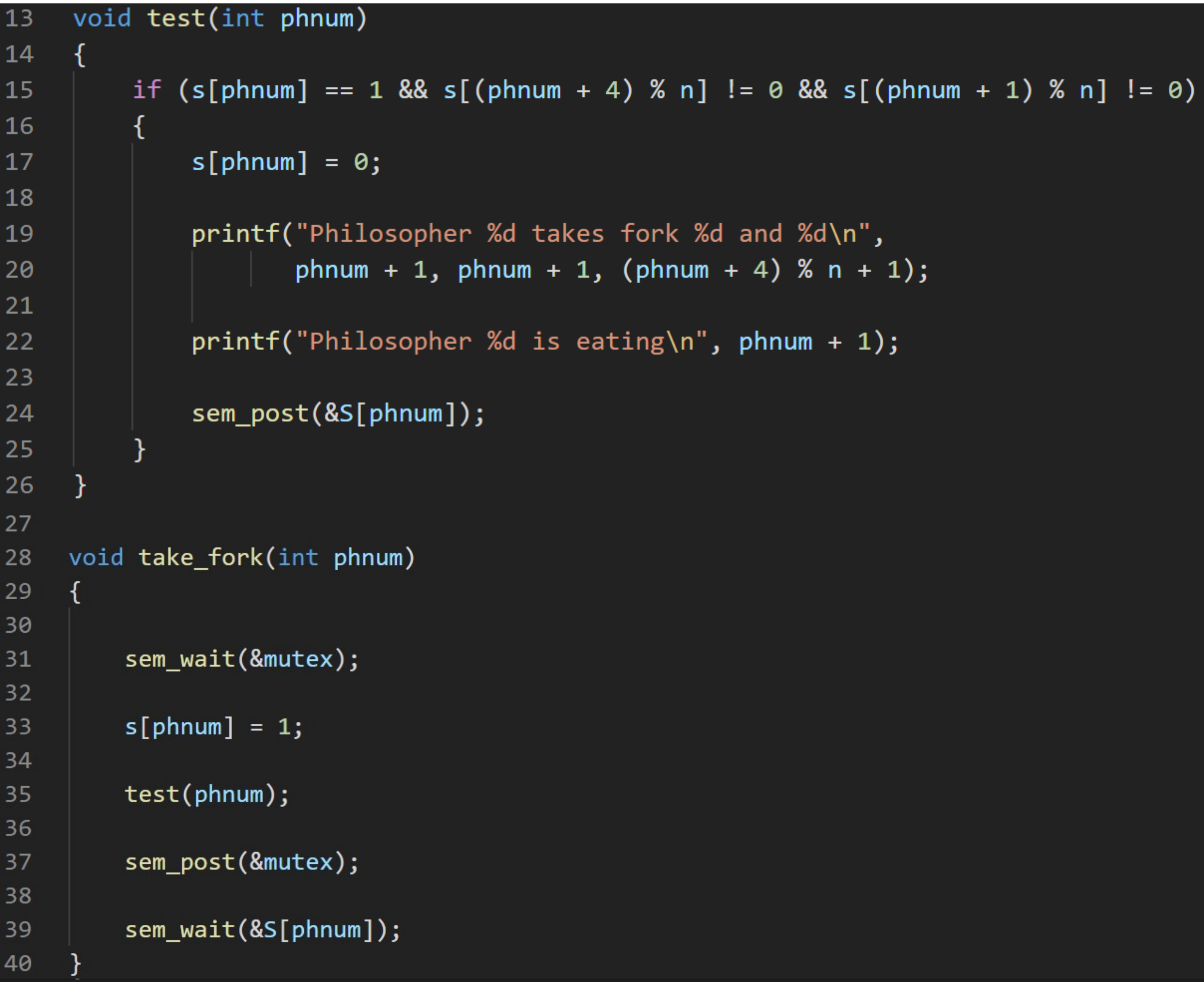
```
void take_fork(int phnum)
{

    sem_wait(&mutex);

    s[phnum] = 1;

    test(phnum);

    sem_post(&mutex);

    sem_wait(&S[phnum]);
}
void put_fork(int phnum)
{

    sem_wait(&mutex);

    s[phnum] = 2;

    printf("Philosopher %d putting fork %d and %d down\n",
           phnum + 1, (phnum + 4) % n + 1, phnum + 1);
    printf("Philosopher %d has finished eating and now thinking again\n", phnum + 1);

    test((phnum + 4) % n);
    test((phnum + 1) % n);

    sem_post(&mutex);
```

```
}

void *philospher(void *num)
{

    int *i = num;

    take_fork(*i);

    put_fork(*i);
}

int main()
{

    int i, *p;
    printf("(19BIT0292)Enter  the total number of philosophers: ");
    scanf("%d", &n);
    pthread_t thread_id[n];
    S = malloc(sizeof(sem_t) * n);
    s = malloc(sizeof(int) * n);
    for (i = 0; i < n; i++)
        p[i] = i;
    sem_init(&mutex, 0, 1);

    for (i = 0; i < n; i++)

        sem_init(&S[i], 0, 0);
    for (i = 0; i < n; i++)
    {
        p = malloc(sizeof(int));
        *p = i + 1;
        pthread_create(&thread_id[i], NULL,
                        philospher, p);

        printf("Philosopher %d is thinking\n", i + 1);
    }

    for (i = 0; i < n; i++)
        pthread_join(thread_id[i], NULL);
}
```

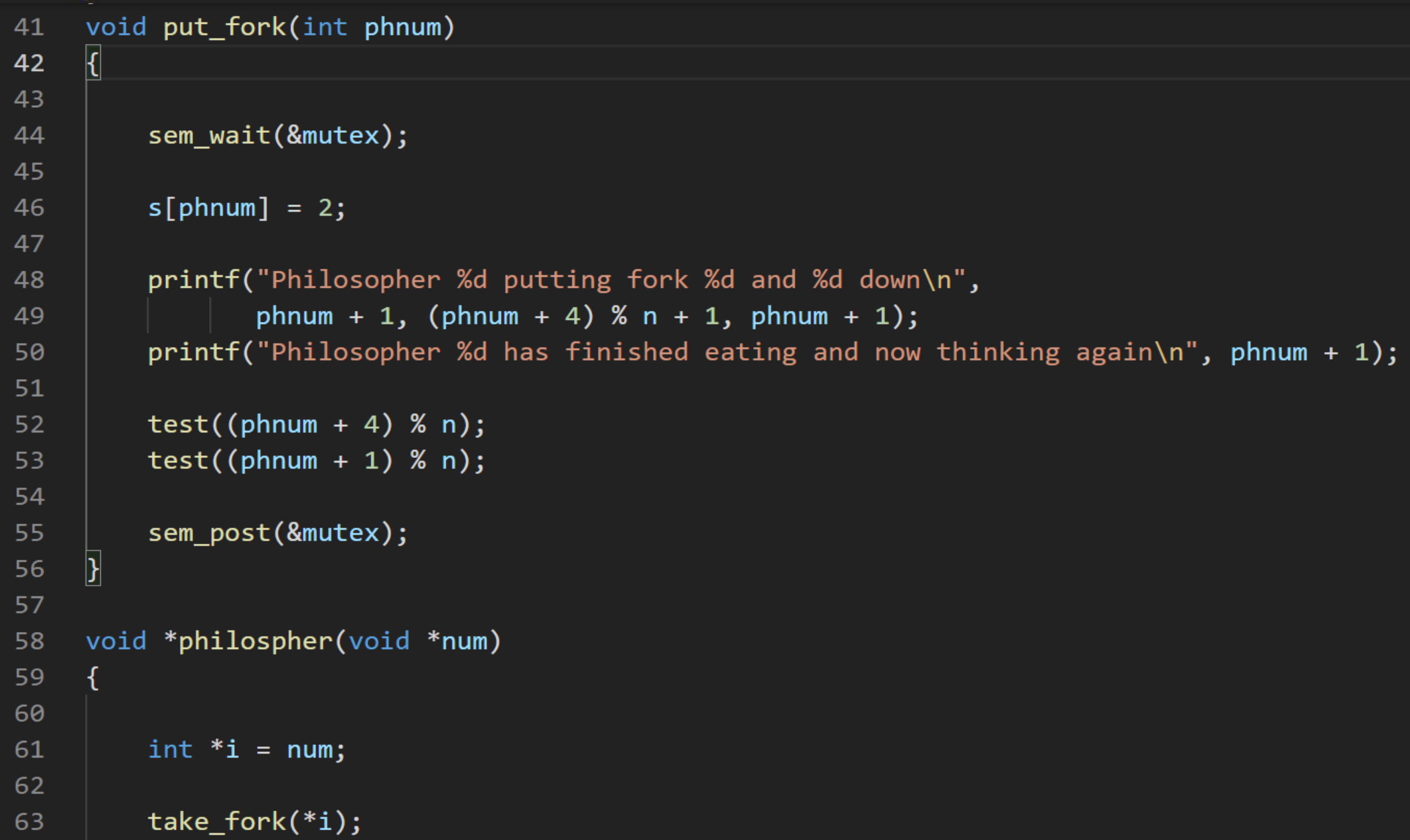
SCREENSHOT OF CODE



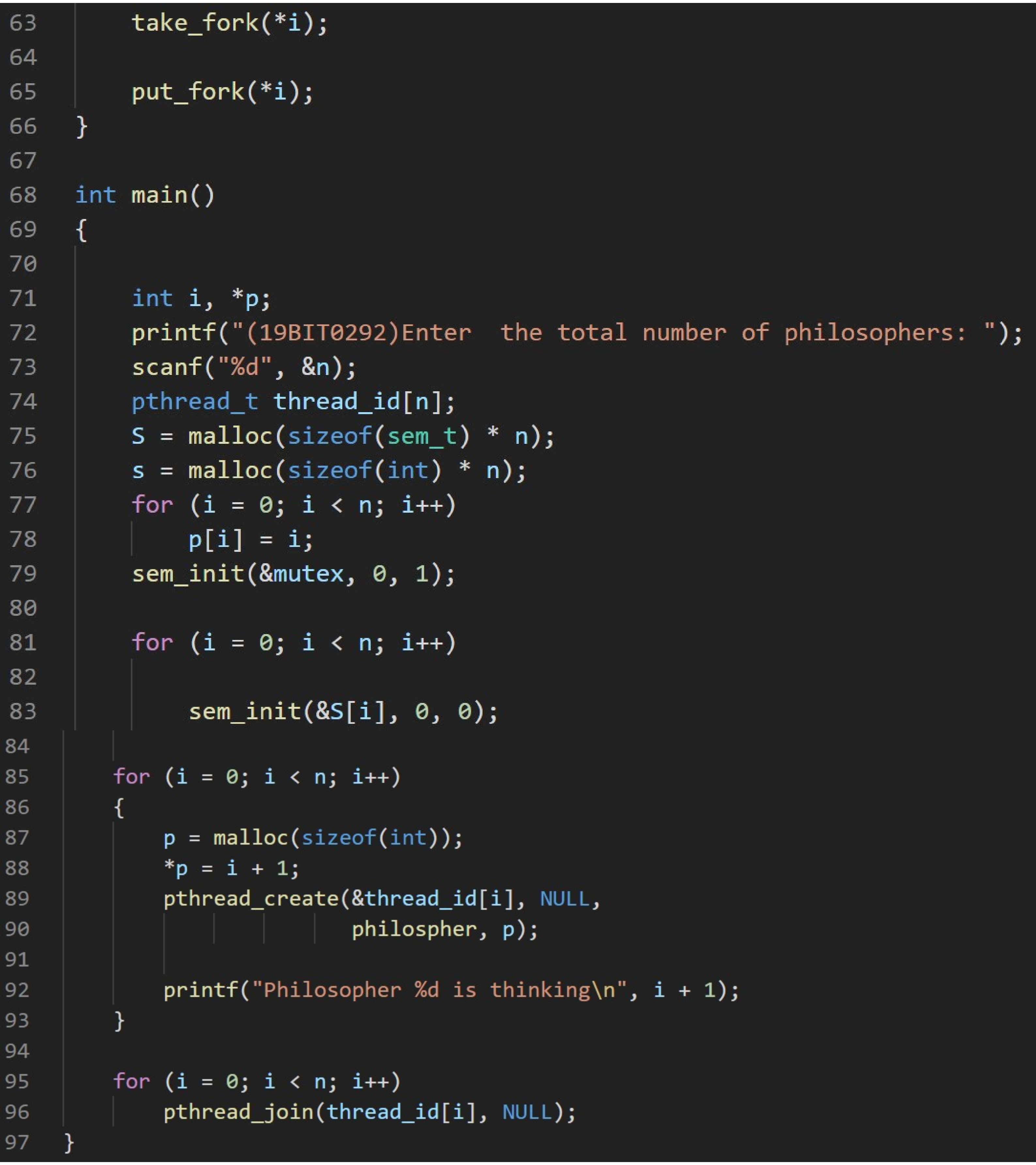
```
#include <pthread.h>
#include <semaphore.h>
#include <stdio.h>
#include <stdlib.h>

int n;

int *s;

sem_t mutex;
sem_t *S;

```



```
void test(int phnum)
{
    if (s[phnum] == 1 && s[(phnum + 4) % n] != 0 && s[(phnum + 1) % n] != 0)
    {
        s[phnum] = 0;

        printf("Philosopher %d takes fork %d and %d\n",
               phnum + 1, phnum + 1, (phnum + 4) % n + 1);

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

        sem_post(&S[phnum]);
    }
}

void take_fork(int phnum)
{

    sem_wait(&mutex);

    s[phnum] = 1;

    test(phnum);

    sem_post(&mutex);

    sem_wait(&S[phnum]);
}
```

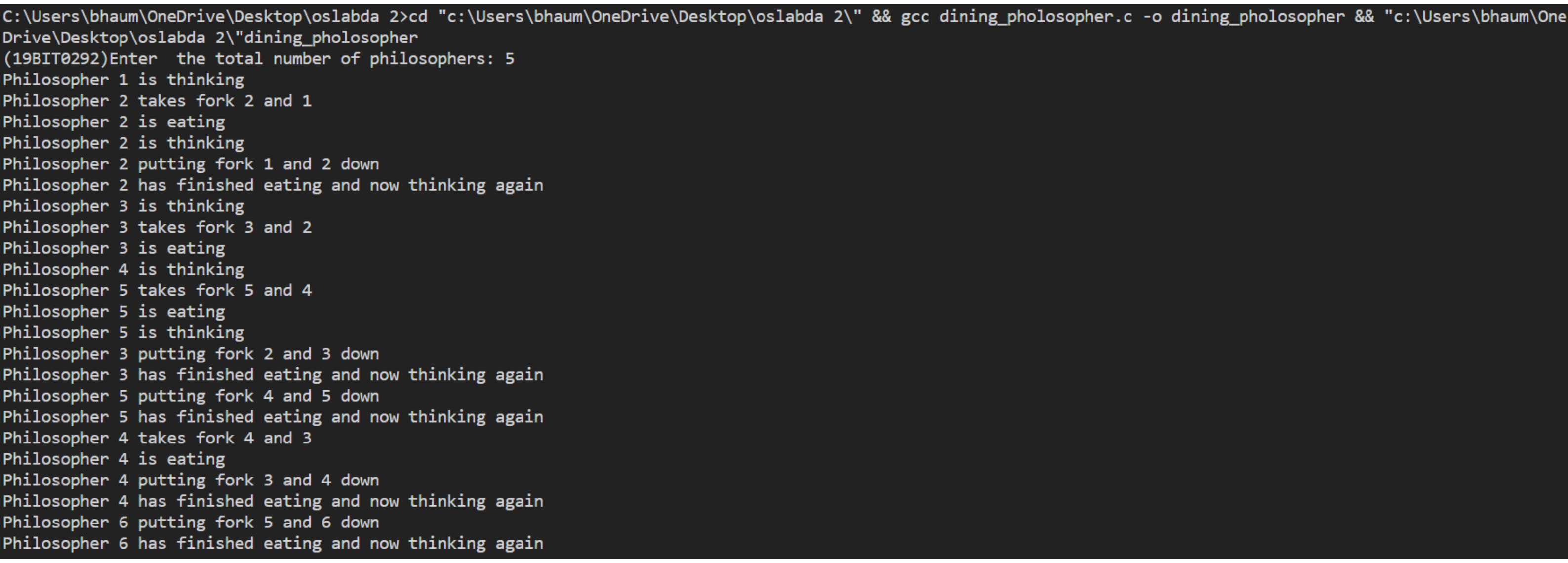


```
void put_fork(int phnum)
{

    sem_wait(&mutex);

    s[phnum] = 2;

    printf("Philosopher %d putting fork %d and %d down\n",
           phnum + 1, (phnum + 4) % n + 1, phnum + 1);
    printf("Philosopher %d has finished eating and now thinking again\n", phnum + 1);

    test((phnum + 4) % n);
    test((phnum + 1) % n);

    sem_post(&mutex);
}

void *philospher(void *num)
{

    int *i = num;

    take_fork(*i);
```

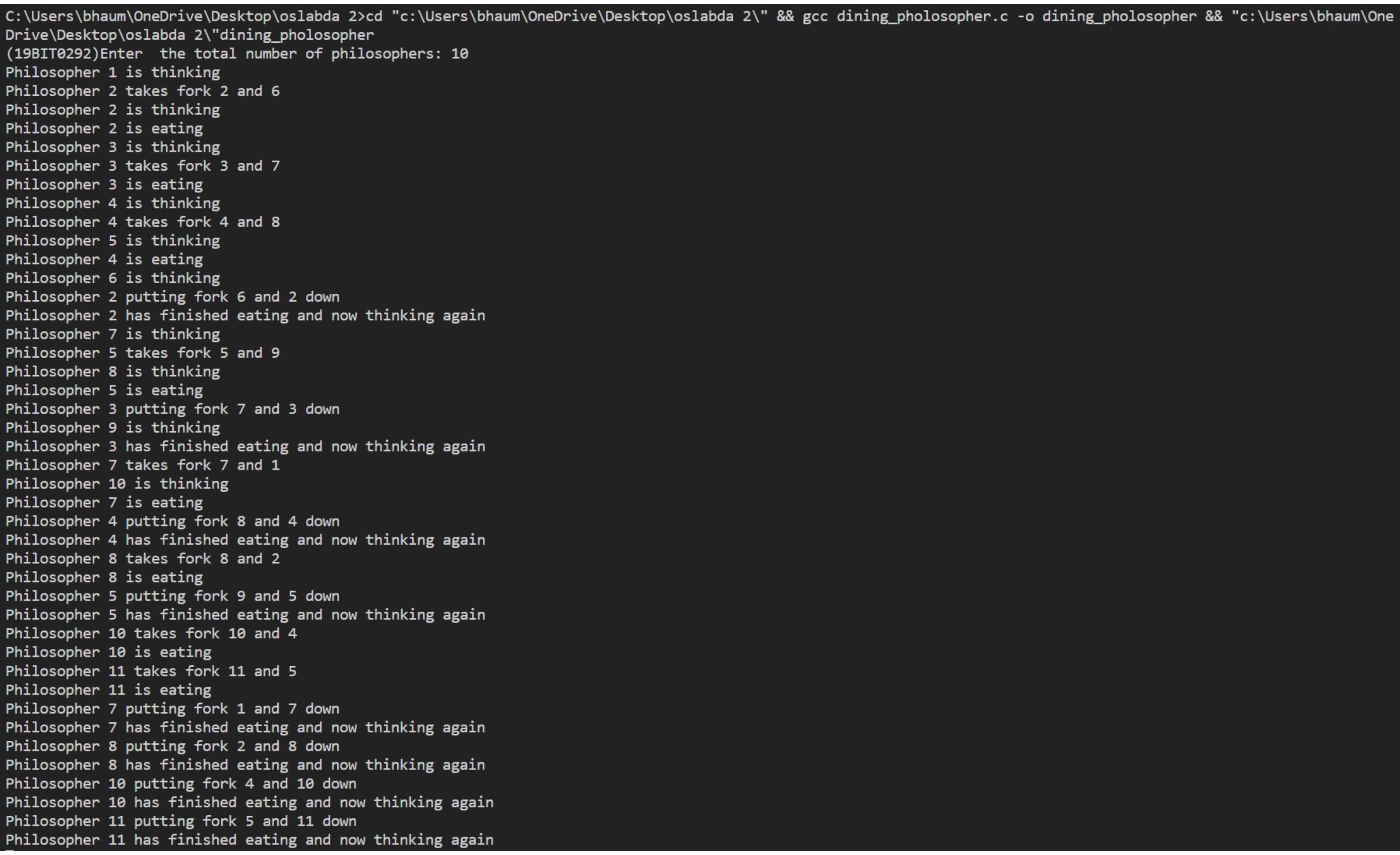


```
    take_fork(*i);

    put_fork(*i);
}

int main()
{

    int i, *p;
    printf("(19BIT0292)Enter  the total number of philosophers: ");
    scanf("%d", &n);
    pthread_t thread_id[n];
    S = malloc(sizeof(sem_t) * n);
    s = malloc(sizeof(int) * n);
    for (i = 0; i < n; i++)
        p[i] = i;
    sem_init(&mutex, 0, 1);

    for (i = 0; i < n; i++)

        sem_init(&S[i], 0, 0);

    for (i = 0; i < n; i++)
    {
        p = malloc(sizeof(int));
        *p = i + 1;
        pthread_create(&thread_id[i], NULL,
                       philospher, p);

        printf("Philosopher %d is thinking\n", i + 1);
    }

    for (i = 0; i < n; i++)
        pthread_join(thread_id[i], NULL);
}
```

OUTPUT



```
C:\Users\bhaum\OneDrive\Desktop\oslabda 2>cd "c:\Users\bhaum\OneDrive\Desktop\oslabda 2\" && gcc dining_pholosopher.c -o dining_pholosopher && "c:\Users\bhaum\One
Drive\Desktop\oslabda 2\"dining_pholosopher
(19BIT0292)Enter  the total number of philosophers: 5
Philosopher 1 is thinking
Philosopher 2 takes fork 2 and 1
Philosopher 2 is eating
Philosopher 2 is thinking
Philosopher 2 putting fork 1 and 2 down
Philosopher 2 has finished eating and now thinking again
Philosopher 3 is thinking
Philosopher 3 takes fork 3 and 2
Philosopher 3 is eating
Philosopher 4 is thinking
Philosopher 5 takes fork 5 and 4
Philosopher 5 is eating
Philosopher 5 is thinking
Philosopher 3 putting fork 2 and 3 down
Philosopher 3 has finished eating and now thinking again
Philosopher 5 putting fork 4 and 5 down
Philosopher 5 has finished eating and now thinking again
Philosopher 4 takes fork 4 and 3
Philosopher 4 is eating
Philosopher 4 putting fork 3 and 4 down
Philosopher 4 has finished eating and now thinking again
Philosopher 6 putting fork 5 and 6 down
Philosopher 6 has finished eating and now thinking again
```



```
C:\Users\bhaum\OneDrive\Desktop\oslabda 2>cd "c:\Users\bhaum\OneDrive\Desktop\oslabda 2\" && gcc dining_pholosopher.c -o dining_pholosopher && "c:\Users\bhaum\One
Drive\Desktop\oslabda 2\"dining_pholosopher
(19BIT0292)Enter  the total number of philosophers: 10
Philosopher 1 is thinking
Philosopher 2 takes fork 2 and 6
Philosopher 2 is thinking
Philosopher 2 is eating
Philosopher 3 is thinking
Philosopher 3 takes fork 3 and 7
Philosopher 3 is eating
Philosopher 4 is thinking
Philosopher 4 takes fork 4 and 8
Philosopher 5 is thinking
Philosopher 4 is eating
Philosopher 6 is thinking
Philosopher 2 putting fork 6 and 2 down
Philosopher 2 has finished eating and now thinking again
Philosopher 7 is thinking
Philosopher 5 takes fork 5 and 9
Philosopher 8 is thinking
Philosopher 5 is eating
Philosopher 3 putting fork 7 and 3 down
Philosopher 9 is thinking
Philosopher 3 has finished eating and now thinking again
Philosopher 7 takes fork 7 and 1
Philosopher 10 is thinking
Philosopher 7 is eating
Philosopher 4 putting fork 8 and 4 down
Philosopher 4 has finished eating and now thinking again
Philosopher 8 takes fork 8 and 2
Philosopher 8 is eating
Philosopher 5 putting fork 9 and 5 down
Philosopher 5 has finished eating and now thinking again
Philosopher 10 takes fork 10 and 4
Philosopher 10 is eating
Philosopher 11 takes fork 11 and 5
Philosopher 11 is eating
Philosopher 7 putting fork 1 and 7 down
Philosopher 7 has finished eating and now thinking again
Philosopher 8 putting fork 2 and 8 down
Philosopher 8 has finished eating and now thinking again
Philosopher 10 putting fork 4 and 10 down
Philosopher 10 has finished eating and now thinking again
Philosopher 11 putting fork 5 and 11 down
Philosopher 11 has finished eating and now thinking again
```

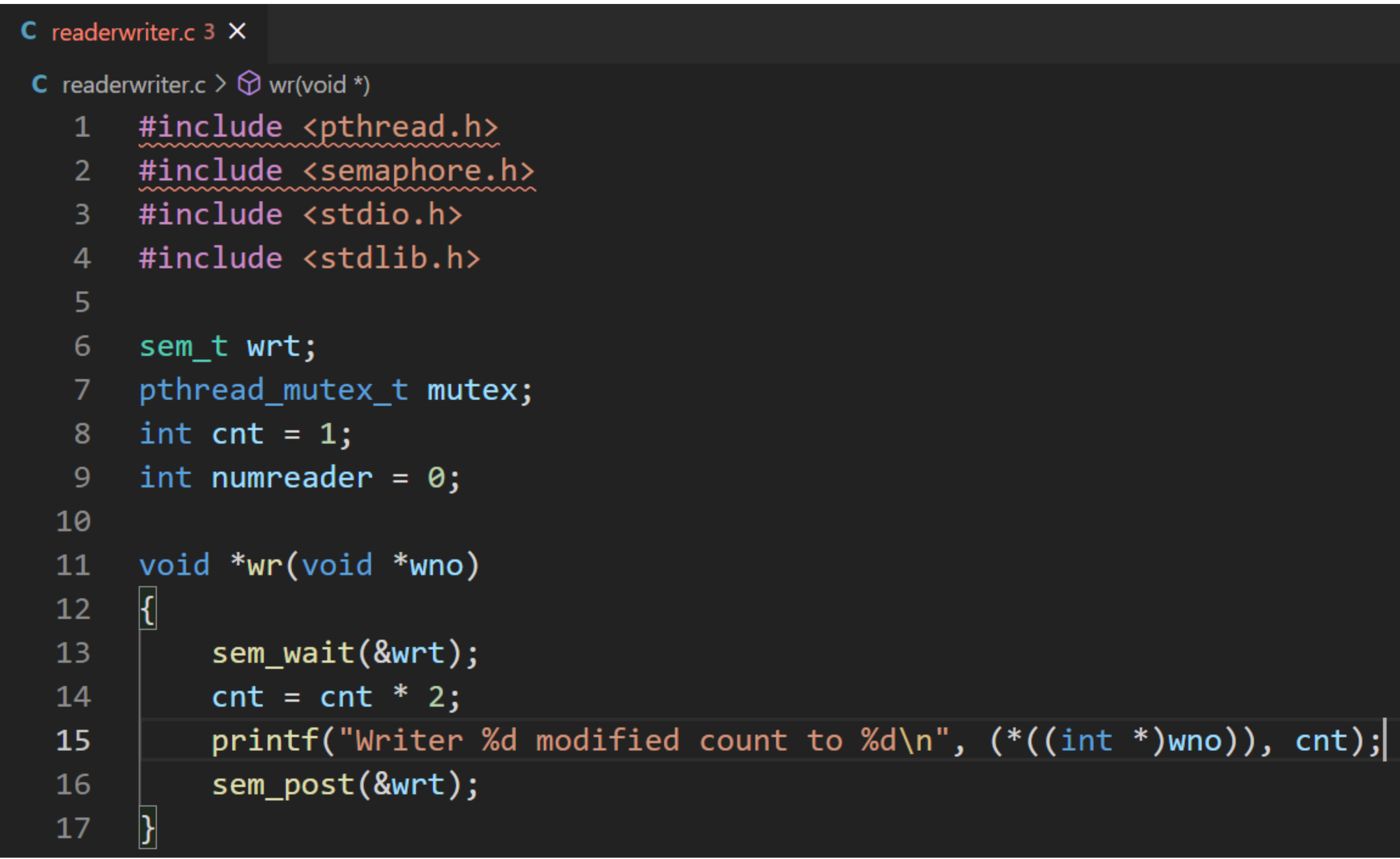
Q3. Write a program to implement the solution for Readers Writers Problem using semaphores.

CODE

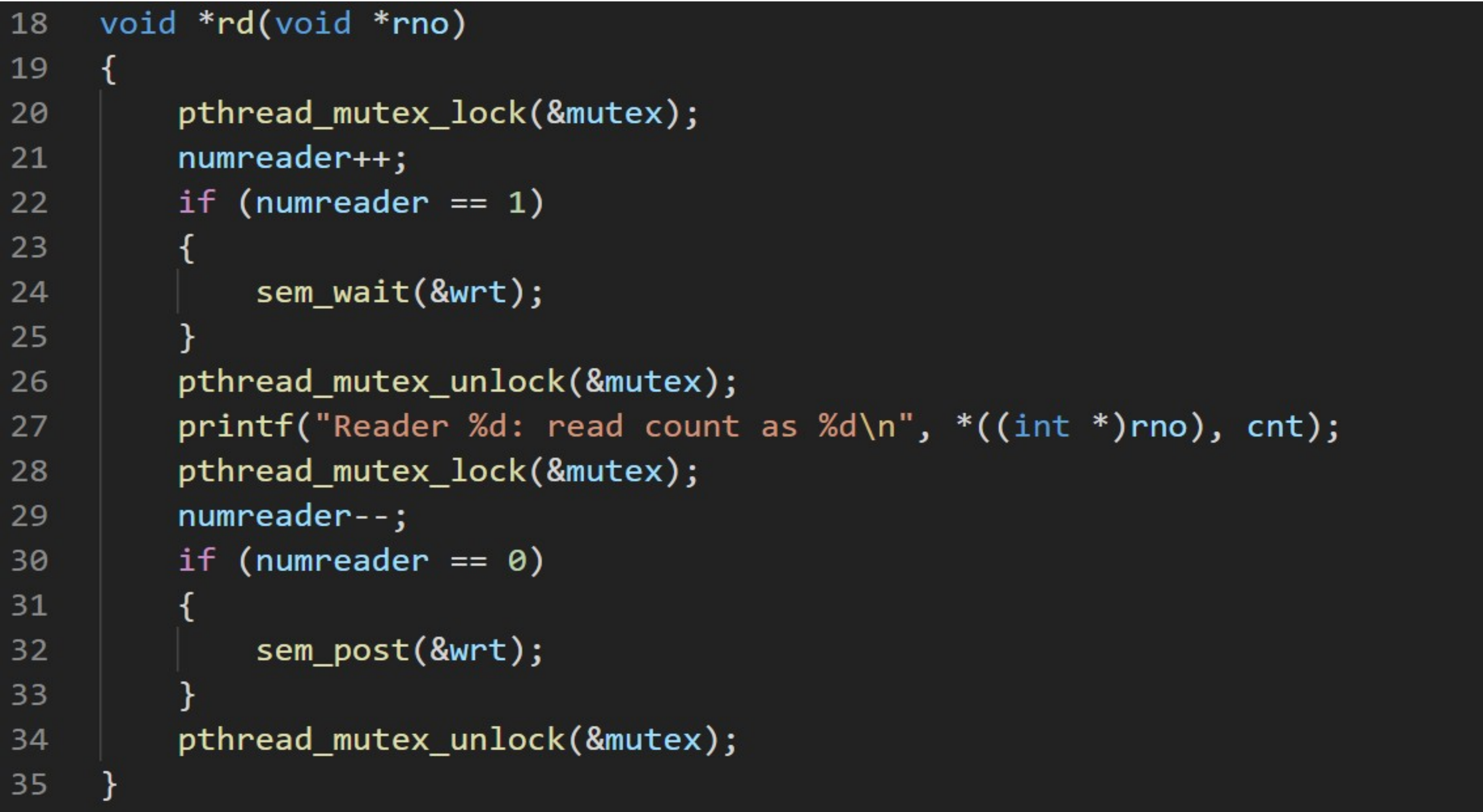
```
#include <pthread.h>

#include <semaphore.h>

#include <stdio.h>

#include <stdlib.h>


sem_t wrt;

pthread_mutex_t mutex;

int cnt = 1;

int numreader = 0;


void *wr(void *wno)

{

    sem_wait(&wrt);

    cnt = cnt * 2;

    printf("Writer %d modified count to %d\n", (*((int *)wno)), cnt);

    sem_post(&wrt);

}

void *rd(void *rno)
```

```
{
    pthread_mutex_lock(&mutex);
    numreader++;
    if (numreader == 1)
    {
        sem_wait(&wrt);
    }
    pthread_mutex_unlock(&mutex);
    printf("Reader %d: read count as %d\n", *((int *)rno), cnt);
    pthread_mutex_lock(&mutex);
    numreader--;
    if (numreader == 0)
    {
        sem_post(&wrt);
    }
    pthread_mutex_unlock(&mutex);
}

void main()
{
    pthread_t *r, *w;
    int rn, wn, *n;
    pthread_mutex_init(&mutex, NULL);
    sem_init(&wrt, 0, 1);
    printf("Enter the number of readers: ");
    scanf("%d", &rn);
    r = (pthread_t *)malloc(sizeof(pthread_t) * rn);
    printf("Enter the number of writers: ");
    scanf("%d", &wn);
    w = (pthread_t *)malloc(sizeof(pthread_t) * wn);
    for (int i = 0; i < rn; i++)
    {
        n = malloc(sizeof(int));
        *n = i + 1;
        pthread_create(&r[i], NULL, (void *)rd, (void *)n);
    }
    for (int i = 0; i < wn; i++)
    {
        n = malloc(sizeof(int));
        *n = i + 1;
        pthread_create(&w[i], NULL, (void *)wr, (void *)n);
    }

    for (int i = 0; i < rn; i++)
        pthread_join(r[i], NULL);
    for (int i = 0; i < wn; i++)
        pthread_join(w[i], NULL);

    pthread_mutex_destroy(&mutex);
    sem_destroy(&wrt);
}
```

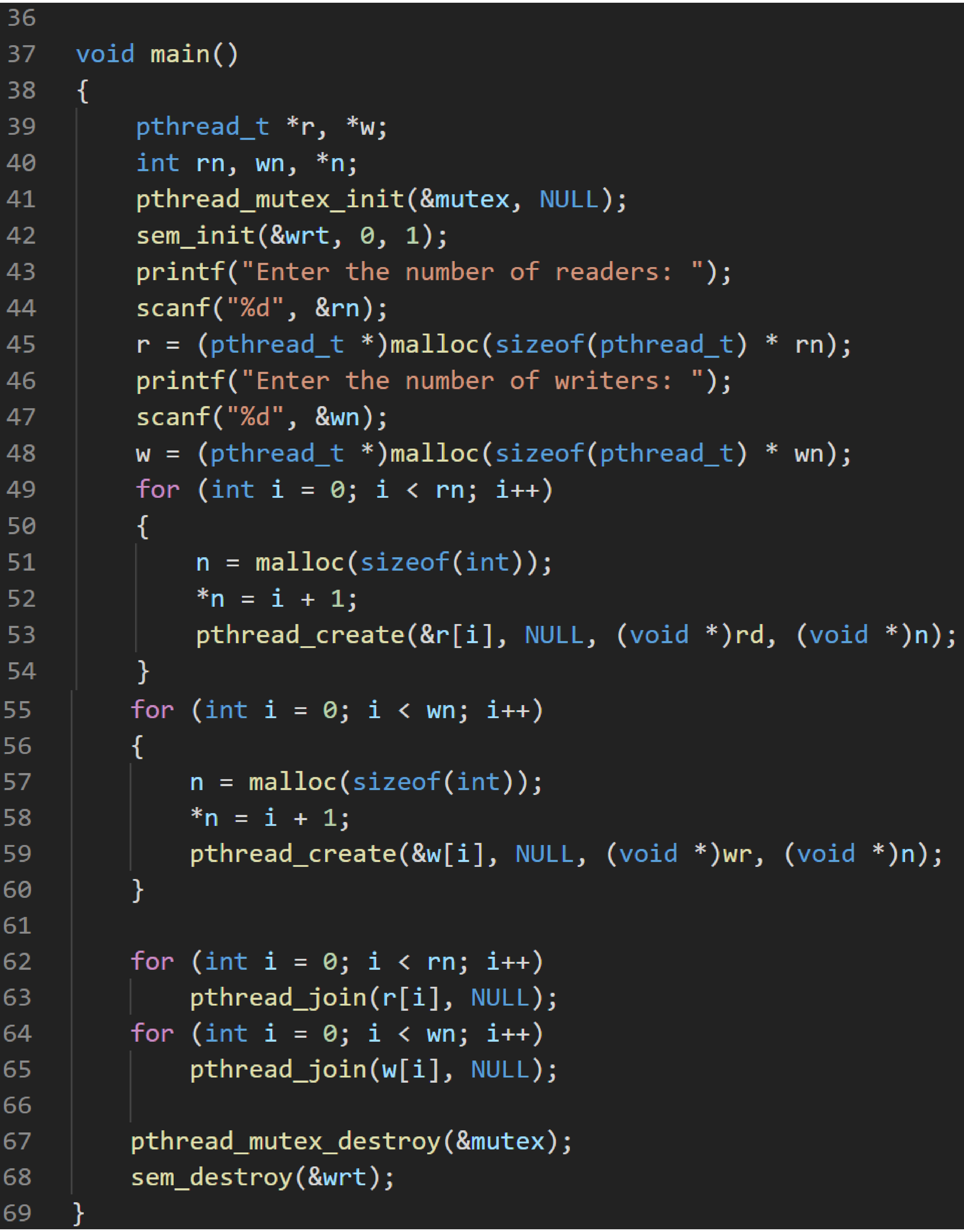
SCREENSHOT OF CODE



```c
#include <pthread.h>
#include <semaphore.h>
#include <stdio.h>
#include <stdlib.h>

sem_t wrt;
pthread_mutex_t mutex;
int cnt = 1;
int numreader = 0;

void *wr(void *wno)
{
    sem_wait(&wrt);
    cnt = cnt * 2;
    printf("Writer %d modified count to %d\n", (*((int *)wno)), cnt);
    sem_post(&wrt);
}
```

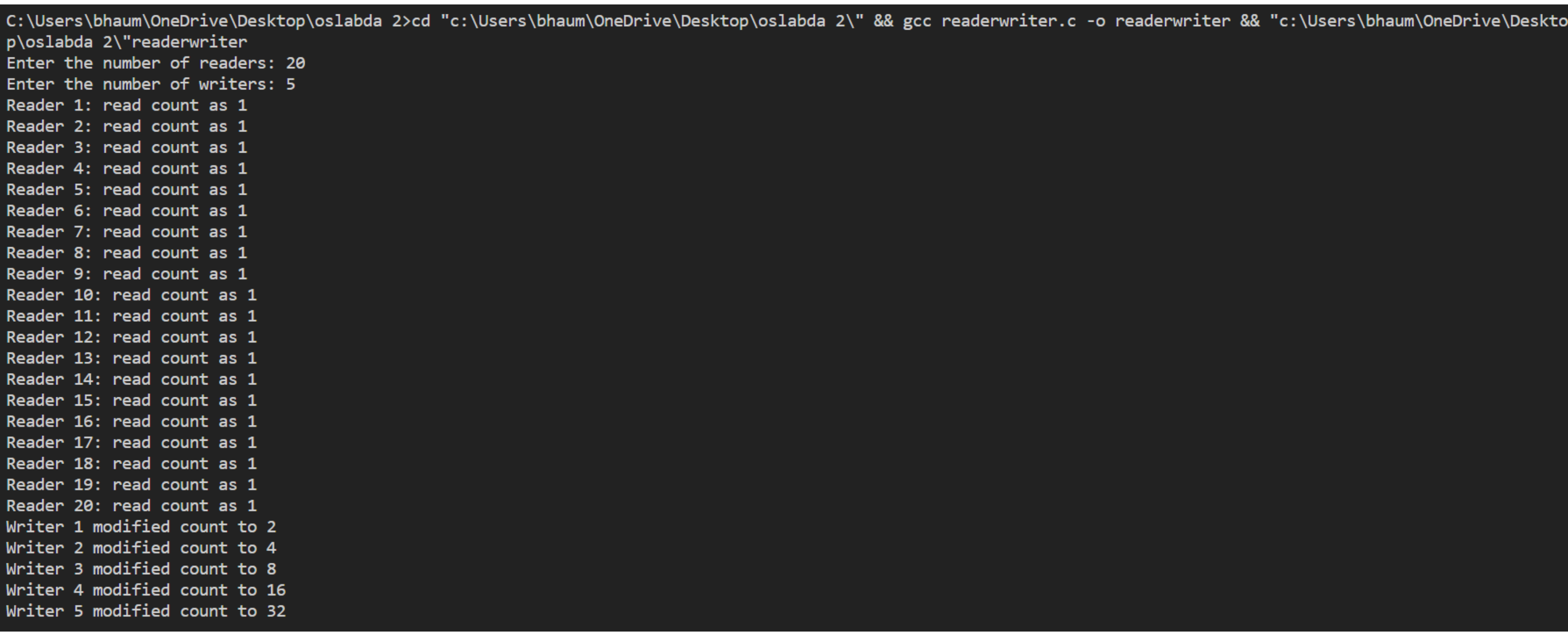


```c
void *rd(void *rno)
{
    pthread_mutex_lock(&mutex);
    numreader++;
    if (numreader == 1)
    {
        sem_wait(&wrt);
    }
    pthread_mutex_unlock(&mutex);
    printf("Reader %d: read count as %d\n", *((int *)rno), cnt);
    pthread_mutex_lock(&mutex);
    numreader--;
    if (numreader == 0)
    {
        sem_post(&wrt);
    }
    pthread_mutex_unlock(&mutex);
}
```

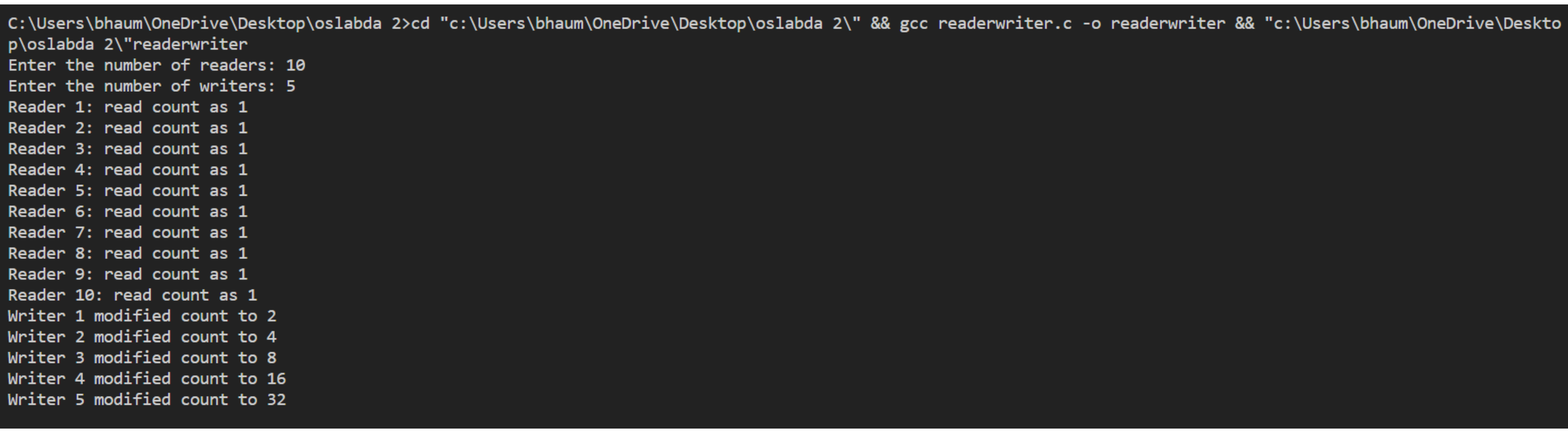


```c

void main()
{
    pthread_t *r, *w;
    int rn, wn, *n;
    pthread_mutex_init(&mutex, NULL);
    sem_init(&wrt, 0, 1);
    printf("Enter the number of readers: ");
    scanf("%d", &rn);
    r = (pthread_t *)malloc(sizeof(pthread_t) * rn);
    printf("Enter the number of writers: ");
    scanf("%d", &wn);
    w = (pthread_t *)malloc(sizeof(pthread_t) * wn);
    for (int i = 0; i < rn; i++)
    {
        n = malloc(sizeof(int));
        *n = i + 1;
        pthread_create(&r[i], NULL, (void *)rd, (void *)n);
    }
    for (int i = 0; i < wn; i++)
    {
        n = malloc(sizeof(int));
        *n = i + 1;
        pthread_create(&w[i], NULL, (void *)wr, (void *)n);
    }

    for (int i = 0; i < rn; i++)
        pthread_join(r[i], NULL);
    for (int i = 0; i < wn; i++)
        pthread_join(w[i], NULL);

    pthread_mutex_destroy(&mutex);
    sem_destroy(&wrt);
}
```

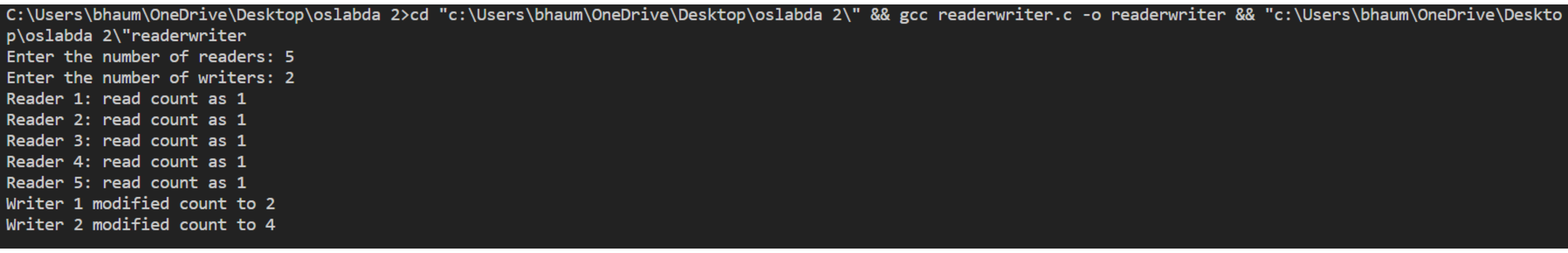
OUTPUT



```
C:\Users\bhaum\OneDrive\Desktop\oslabda 2>cd "c:\Users\bhaum\OneDrive\Desktop\oslabda 2\" && gcc readerwriter.c -o readerwriter && "c:\Users\bhaum\OneDrive\Deskto
p\oslabda 2\"readerwriter
Enter the number of readers: 20
Enter the number of writers: 5
Reader 1: read count as 1
Reader 2: read count as 1
Reader 3: read count as 1
Reader 4: read count as 1
Reader 5: read count as 1
Reader 6: read count as 1
Reader 7: read count as 1
Reader 8: read count as 1
Reader 9: read count as 1
Reader 10: read count as 1
Reader 11: read count as 1
Reader 12: read count as 1
Reader 13: read count as 1
Reader 14: read count as 1
Reader 15: read count as 1
Reader 16: read count as 1
Reader 17: read count as 1
Reader 18: read count as 1
Reader 19: read count as 1
Reader 20: read count as 1
Writer 1 modified count to 2
Writer 2 modified count to 4
Writer 3 modified count to 8
Writer 4 modified count to 16
Writer 5 modified count to 32
```



```
C:\Users\bhaum\OneDrive\Desktop\oslabda 2>cd "c:\Users\bhaum\OneDrive\Desktop\oslabda 2\" && gcc readerwriter.c -o readerwriter && "c:\Users\bhaum\OneDrive\Deskto
p\oslabda 2\"readerwriter
Enter the number of readers: 10
Enter the number of writers: 5
Reader 1: read count as 1
Reader 2: read count as 1
Reader 3: read count as 1
Reader 4: read count as 1
Reader 5: read count as 1
Reader 6: read count as 1
Reader 7: read count as 1
Reader 8: read count as 1
Reader 9: read count as 1
Reader 10: read count as 1
Writer 1 modified count to 2
Writer 2 modified count to 4
Writer 3 modified count to 8
Writer 4 modified count to 16
Writer 5 modified count to 32
```



```
C:\Users\bhaum\OneDrive\Desktop\oslabda 2>cd "c:\Users\bhaum\OneDrive\Desktop\oslabda 2\" && gcc readerwriter.c -o readerwriter && "c:\Users\bhaum\OneDrive\Deskto
p\oslabda 2\"readerwriter
Enter the number of readers: 5
Enter the number of writers: 2
Reader 1: read count as 1
Reader 2: read count as 1
Reader 3: read count as 1
Reader 4: read count as 1
Reader 5: read count as 1
Writer 1 modified count to 2
Writer 2 modified count to 4
```

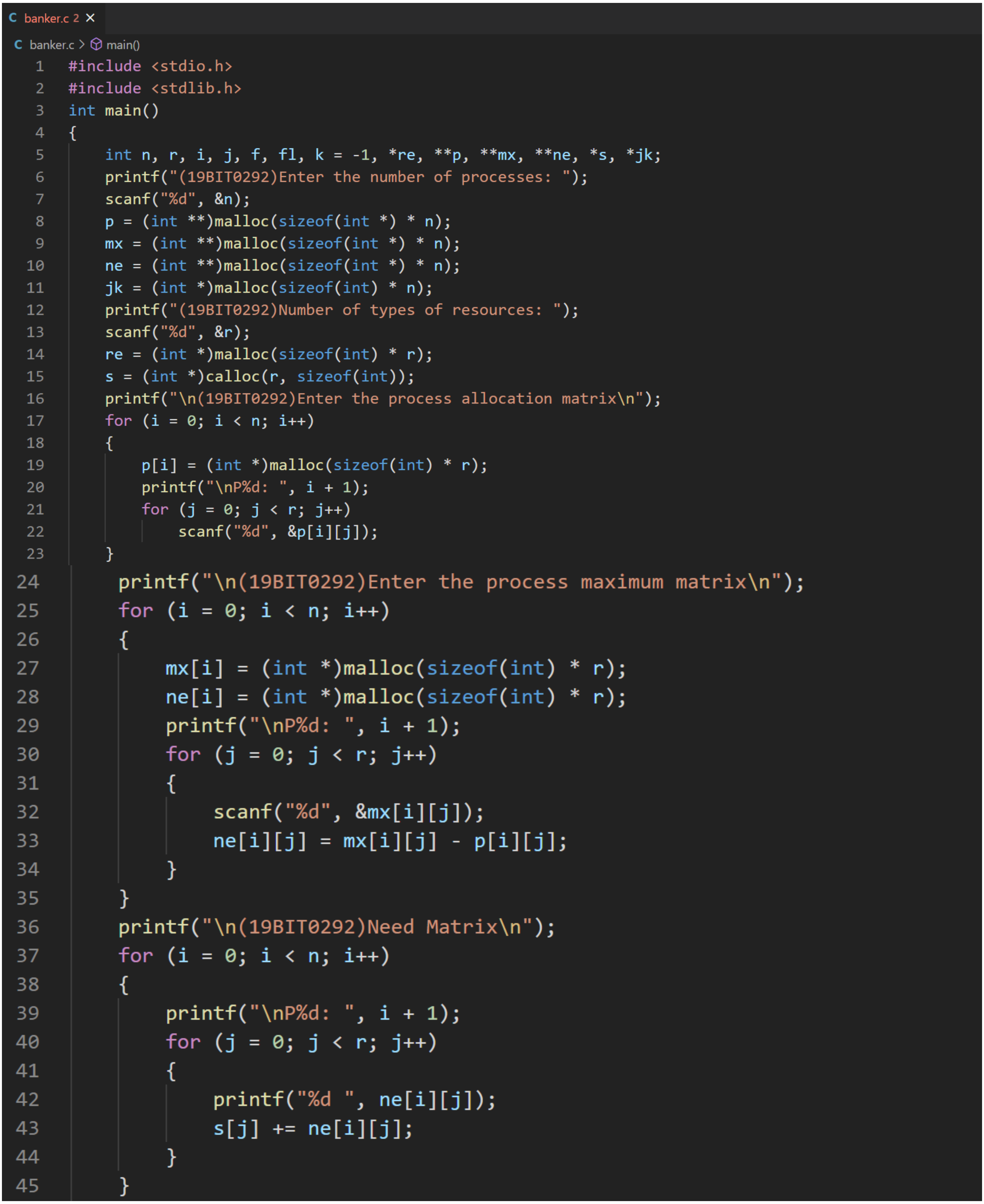
Q4. Write a Program to implement banker’s algorithm for Deadlock avoidance.

CODE

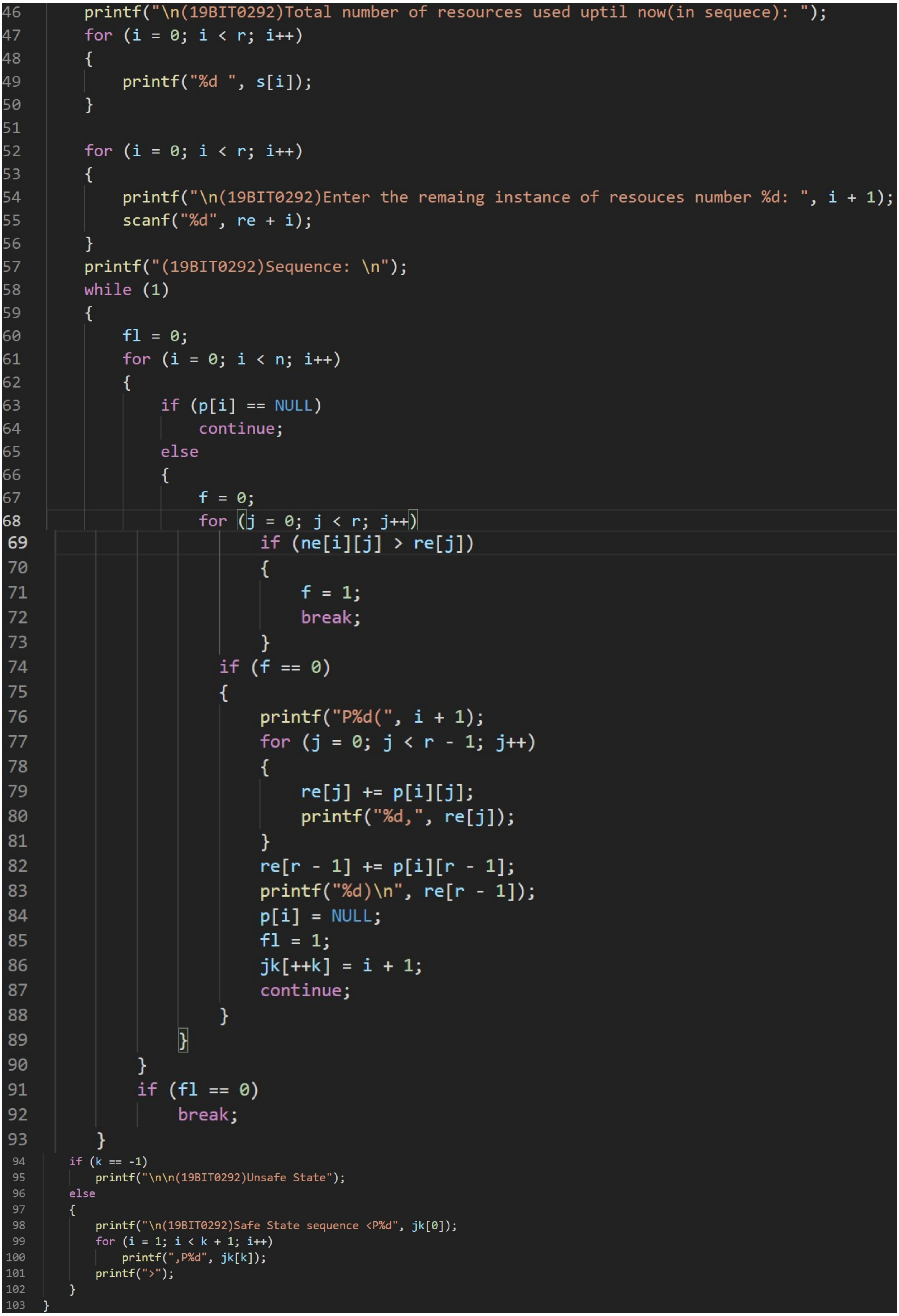
```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int n, r, i, j, f, fl, k = -1, *re, **p, **mx, **ne, *s, *jk;
    printf("(19BIT0292)Enter the number of processes: ");
    scanf("%d", &n);
    p = (int **)malloc(sizeof(int *) * n);
    mx = (int **)malloc(sizeof(int *) * n);
    ne = (int **)malloc(sizeof(int *) * n);
    jk = (int *)malloc(sizeof(int) * n);
    printf("(19BIT0292)Number of types of resources: ");
    scanf("%d", &r);
    re = (int *)malloc(sizeof(int) * r);
    s = (int *)calloc(r, sizeof(int));
    printf("\n(19BIT0292)Enter the process allocation matrix\n");
    for (i = 0; i < n; i++)
    {
        p[i] = (int *)malloc(sizeof(int) * r);
        printf("\nP%d: ", i + 1);
        for (j = 0; j < r; j++)
            scanf("%d", &p[i][j]);
    }
    printf("\n(19BIT0292)Enter the process maximum matrix\n");
    for (i = 0; i < n; i++)
```

```
{
    mx[i] = (int *)malloc(sizeof(int) * r);
    ne[i] = (int *)malloc(sizeof(int) * r);
    printf("\nP%d: ", i + 1);
    for (j = 0; j < r; j++)
    {
        scanf("%d", &mx[i][j]);
        ne[i][j] = mx[i][j] - p[i][j];
    }
}
printf("\n(19BIT0292)Need Matrix\n");
for (i = 0; i < n; i++)
{
    printf("\nP%d: ", i + 1);
    for (j = 0; j < r; j++)
    {
        printf("%d ", ne[i][j]);
        s[j] += ne[i][j];
    }
}
printf("\n(19BIT0292)Total number of resources used uptil now(in sequece): ");
for (i = 0; i < r; i++)
{
    printf("%d ", s[i]);
}

for (i = 0; i < r; i++)
{
    printf("\n(19BIT0292)Enter the remaing instance of resouces number %d: ", i + 1);
    scanf("%d", re + i);
}
printf("(19BIT0292)Sequence: \n");
while (1)
{
    fl = 0;
    for (i = 0; i < n; i++)
    {
        if (p[i] == NULL)
            continue;
            else
            {
                f = 0;
                for (j = 0; j < r; j++)
                    if (ne[i][j] > re[j])
                    {
                        f = 1;
                        break;
                    }
                if (f == 0)
                {
                    printf("P%d(", i + 1);
                    for (j = 0; j < r - 1; j++)
                    {
                        re[j] += p[i][j];
                        printf("%d,", re[j]);
                    }
                    re[r - 1] += p[i][r - 1];
                    printf("%d)\n", re[r - 1]);
                    p[i] = NULL;
                    fl = 1;
                    jk[++k] = i + 1;
                    continue;
                }
            }
        }
        if (fl == 0)
            break;
    }
    if (k == -1)
        printf("\n\n(19BIT0292)Unsafe State");
    else
    {
        printf("\n(19BIT0292)Safe State sequence <P%d", jk[0]);
        for (i = 1; i < k + 1; i++)
            printf(",P%d", jk[k]);
        printf(">");
    }
}
```

SCREENSHOT OF CODE

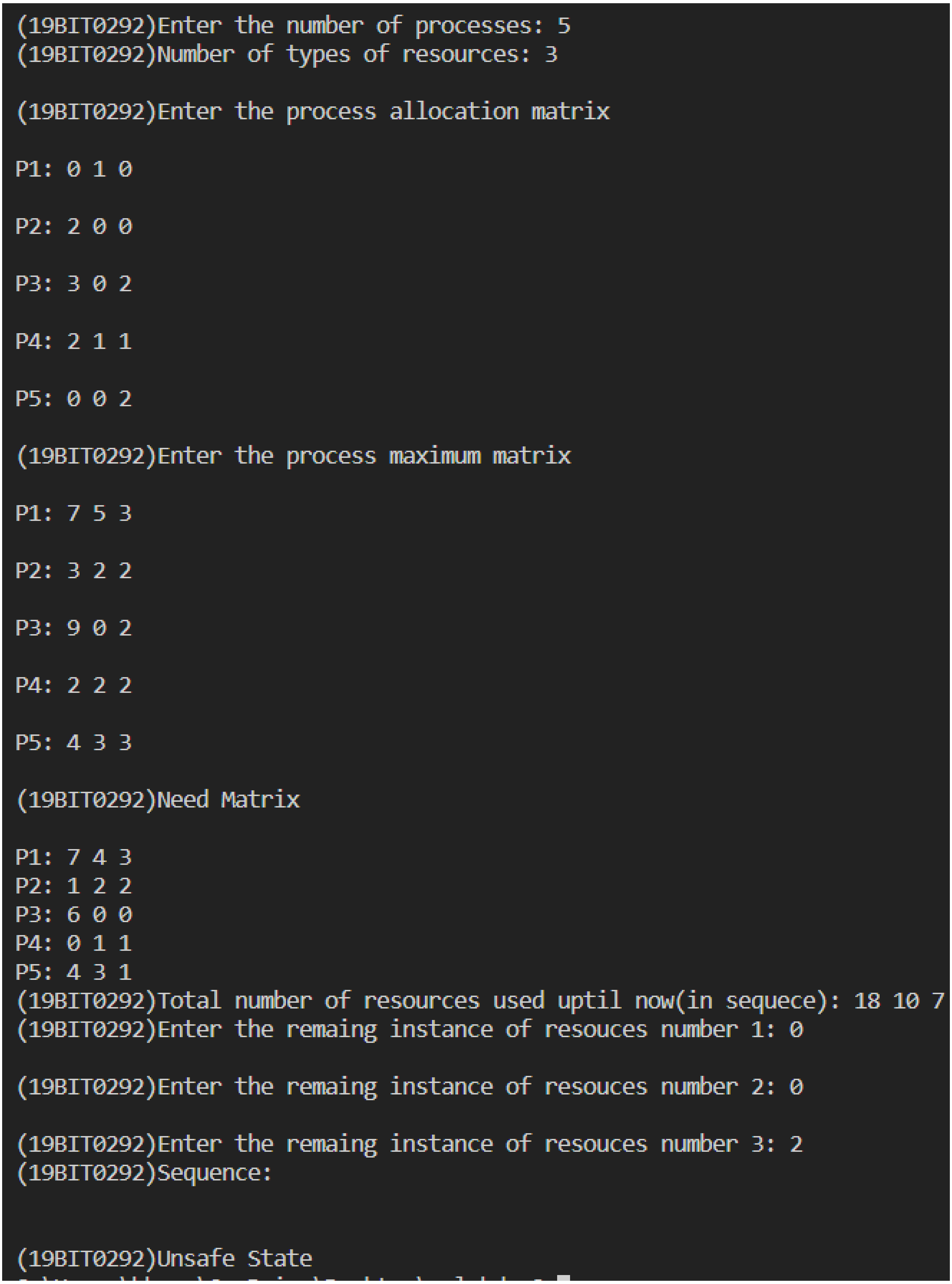


```c
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int n, r, i, j, f, fl, k = -1, *re, **p, **mx, **ne, *s, *jk;
    printf("(19BIT0292)Enter the number of processes: ");
    scanf("%d", &n);
    p = (int **)malloc(sizeof(int *) * n);
    mx = (int **)malloc(sizeof(int *) * n);
    ne = (int **)malloc(sizeof(int *) * n);
    jk = (int *)malloc(sizeof(int) * n);
    printf("(19BIT0292)Number of types of resources: ");
    scanf("%d", &r);
    re = (int *)malloc(sizeof(int) * r);
    s = (int *)calloc(r, sizeof(int));
    printf("\n(19BIT0292)Enter the process allocation matrix\n");
    for (i = 0; i < n; i++)
    {
        p[i] = (int *)malloc(sizeof(int) * r);
        printf("\nP%d: ", i + 1);
        for (j = 0; j < r; j++)
            scanf("%d", &p[i][j]);
    }
    printf("\n(19BIT0292)Enter the process maximum matrix\n");
    for (i = 0; i < n; i++)
    {
        mx[i] = (int *)malloc(sizeof(int) * r);
        ne[i] = (int *)malloc(sizeof(int) * r);
        printf("\nP%d: ", i + 1);
        for (j = 0; j < r; j++)
        {
            scanf("%d", &mx[i][j]);
            ne[i][j] = mx[i][j] - p[i][j];
        }
    }
    printf("\n(19BIT0292)Need Matrix\n");
    for (i = 0; i < n; i++)
    {
        printf("\nP%d: ", i + 1);
        for (j = 0; j < r; j++)
        {
            printf("%d ", ne[i][j]);
            s[j] += ne[i][j];
        }
    }
```

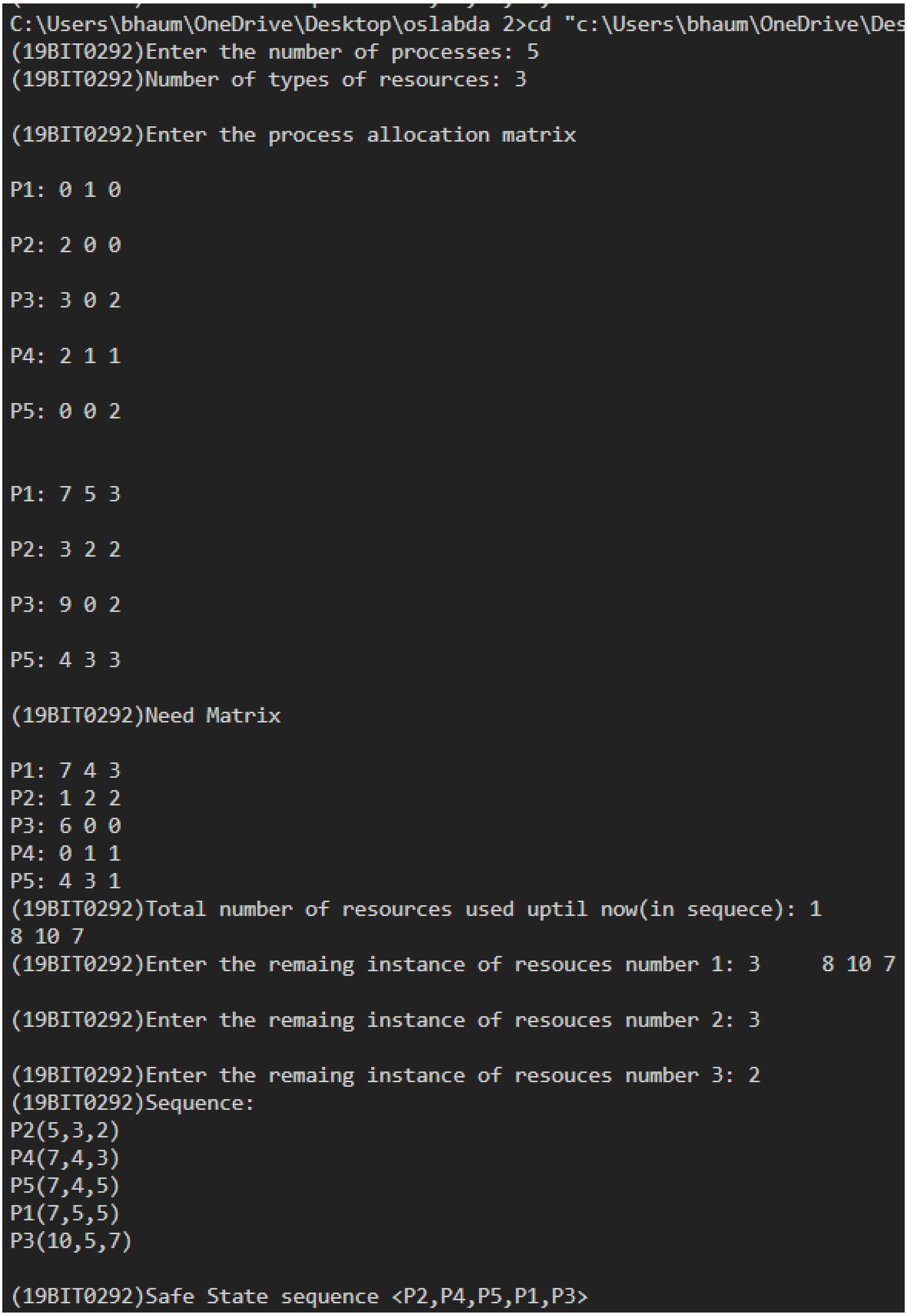


```c
    printf("\n(19BIT0292)Total number of resources used uptil now(in sequece): ");
    for (i = 0; i < r; i++)
    {
        printf("%d ", s[i]);
    }

    for (i = 0; i < r; i++)
    {
        printf("\n(19BIT0292)Enter the remaing instance of resouces number %d: ", i + 1);
        scanf("%d", re + i);
    }
    printf("(19BIT0292)Sequence: \n");
    while (1)
    {
        fl = 0;
        for (i = 0; i < n; i++)
        {
            if (p[i] == NULL)
                continue;
            else
            {
                f = 0;
                for (j = 0; j < r; j++)
                    if (ne[i][j] > re[j])
                    {
                        f = 1;
                        break;
                    }
                if (f == 0)
                {
                    printf("P%d(", i + 1);
                    for (j = 0; j < r - 1; j++)
                    {
                        re[j] += p[i][j];
                        printf("%d,", re[j]);
                    }
                    re[r - 1] += p[i][r - 1];
                    printf("%d)\n", re[r - 1]);
                    p[i] = NULL;
                    fl = 1;
                    jk[++k] = i + 1;
                    continue;
                }
            }
        }
        if (fl == 0)
            break;
    }
    if (k == -1)
        printf("\n\n(19BIT0292)Unsafe State");
    else
    {
        printf("\n(19BIT0292)Safe State sequence <P%d", jk[0]);
        for (i = 1; i < k + 1; i++)
            printf(",P%d", jk[k]);
        printf(">");
    }
}
```

OUTPUT



```
(19BIT0292)Enter the number of processes: 5
(19BIT0292)Number of types of resources: 3

(19BIT0292)Enter the process allocation matrix

P1: 0 1 0

P2: 2 0 0

P3: 3 0 2

P4: 2 1 1

P5: 0 0 2

(19BIT0292)Enter the process maximum matrix

P1: 7 5 3

P2: 3 2 2

P3: 9 0 2

P4: 2 2 2

P5: 4 3 3

(19BIT0292)Need Matrix

P1: 7 4 3
P2: 1 2 2
P3: 6 0 0
P4: 0 1 1
P5: 4 3 1
(19BIT0292)Total number of resources used uptil now(in sequece): 18 10 7
(19BIT0292)Enter the remaing instance of resouces number 1: 0

(19BIT0292)Enter the remaing instance of resouces number 2: 0

(19BIT0292)Enter the remaing instance of resouces number 3: 2
(19BIT0292)Sequence:


(19BIT0292)Unsafe State
```



```
C:\Users\bhaum\OneDrive\Desktop\oslabda 2>cd "c:\Users\bhaum\OneDrive\Des
(19BIT0292)Enter the number of processes: 5
(19BIT0292)Number of types of resources: 3

(19BIT0292)Enter the process allocation matrix

P1: 0 1 0

P2: 2 0 0

P3: 3 0 2

P4: 2 1 1

P5: 0 0 2


P1: 7 5 3

P2: 3 2 2

P3: 9 0 2

P5: 4 3 3

(19BIT0292)Need Matrix

P1: 7 4 3
P2: 1 2 2
P3: 6 0 0
P4: 0 1 1
P5: 4 3 1
(19BIT0292)Total number of resources used uptil now(in sequece): 1
8 10 7
(19BIT0292)Enter the remaing instance of resouces number 1: 3     8 10 7

(19BIT0292)Enter the remaing instance of resouces number 2: 3

(19BIT0292)Enter the remaing instance of resouces number 3: 2
(19BIT0292)Sequence:
P2(5,3,2)
P4(7,4,3)
P5(7,4,5)
P1(7,5,5)
P3(10,5,7)

(19BIT0292)Safe State sequence <P2,P4,P5,P1,P3>
```

CLICK HERE FOR GITHUBB LINK