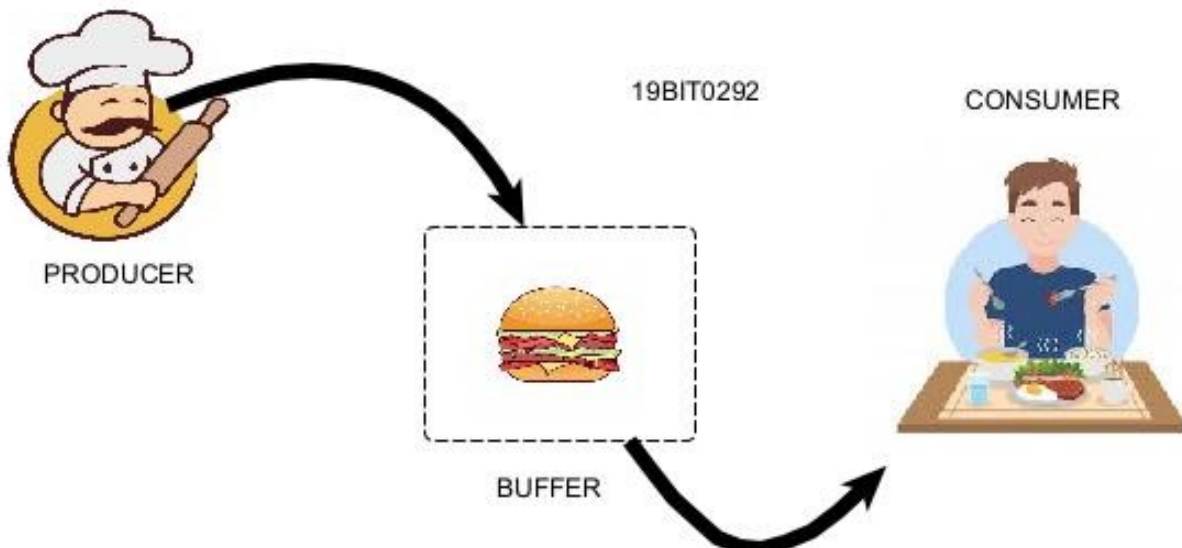


**19BIT0292**  
**Bhaumik Tandan**

**ASSESSMENT-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 producer\_consumer.c 3 X

C producer\_consumer.c > main()

```
1  #include <pthread.h>
2  #include <semaphore.h>
3  #include <stdlib.h>
4  #include <stdio.h>
5
6  int max;
7  int m, b;
8  sem_t e;
9  sem_t f;
10 int in = 0;
11 int out = 0;
12 int *buffer;
13 pthread_mutex_t mx;
14
15 void *pro(void *pno)
16 {
17     int j;
18     for (int i = 0; i < b; i++)
19     {
20         j = rand() % 100;
21         sem_wait(&e);
22         printf("Producer %d: Insert Item %d at %d\n", *((int *)pno), buffer[in], in);
23         in = (in + 1) % b;
24         pthread_mutex_unlock(&mx);
25         sem_post(&f);
26     }
27 }
28
29 void *con(void *cno)
30 {
31     for (int i = 0; i < b; i++)
32     {
33         sem_wait(&f);
34         pthread_mutex_lock(&mx);
35         int j = buffer[out];
36         printf("Consumer %d: Remove Item %d from %d\n", *((int *)cno), j, out);
37         out = (out + 1) % b;
38         pthread_mutex_unlock(&mx);
39         sem_post(&e);
40     }
41 }
42
43
```

```

44 void main()
45 {
46     pthread_t *p, *c;
47     int rn, wn, *n;
48     pthread_mutex_init(&mx, NULL);
49     printf("(19BIT0292)Enter the size of the buffer: ");
50     scanf("%d", &b);
51     buffer = (int *)malloc(sizeof(int) * b);
52     sem_init(&e, 0, b);
53     sem_init(&f, 0, 0);
54     printf("Enter the number of producers: ");
55     scanf("%d", &rn);
56     p = (pthread_t *)malloc(sizeof(pthread_t) * rn);
57     printf("Enter the number of consumers: ");
58     scanf("%d", &wn);
59     c = (pthread_t *)malloc(sizeof(pthread_t) * wn);
60     for (int i = 0; i < rn; i++)
61     {
62         n = malloc(sizeof(int));
63         *n = i + 1;
64         pthread_create(&p[i], NULL, pro, n);
65         n = NULL;
66     }
67     for (int i = 0; i < wn; i++)
68     {
69         n = malloc(sizeof(int));
70         *n = i + 1;
71         pthread_create(&c[i], NULL, con, n);
72         n = NULL;
73     }
74
75     for (int i = 0; i < rn; i++)
76         pthread_join(p[i], NULL);
77     for (int i = 0; i < wn; i++)
78         pthread_join(c[i], NULL);
79
80     pthread_mutex_destroy(&mx);
81     sem_destroy(&e);
82     sem_destroy(&f);
83 }

```

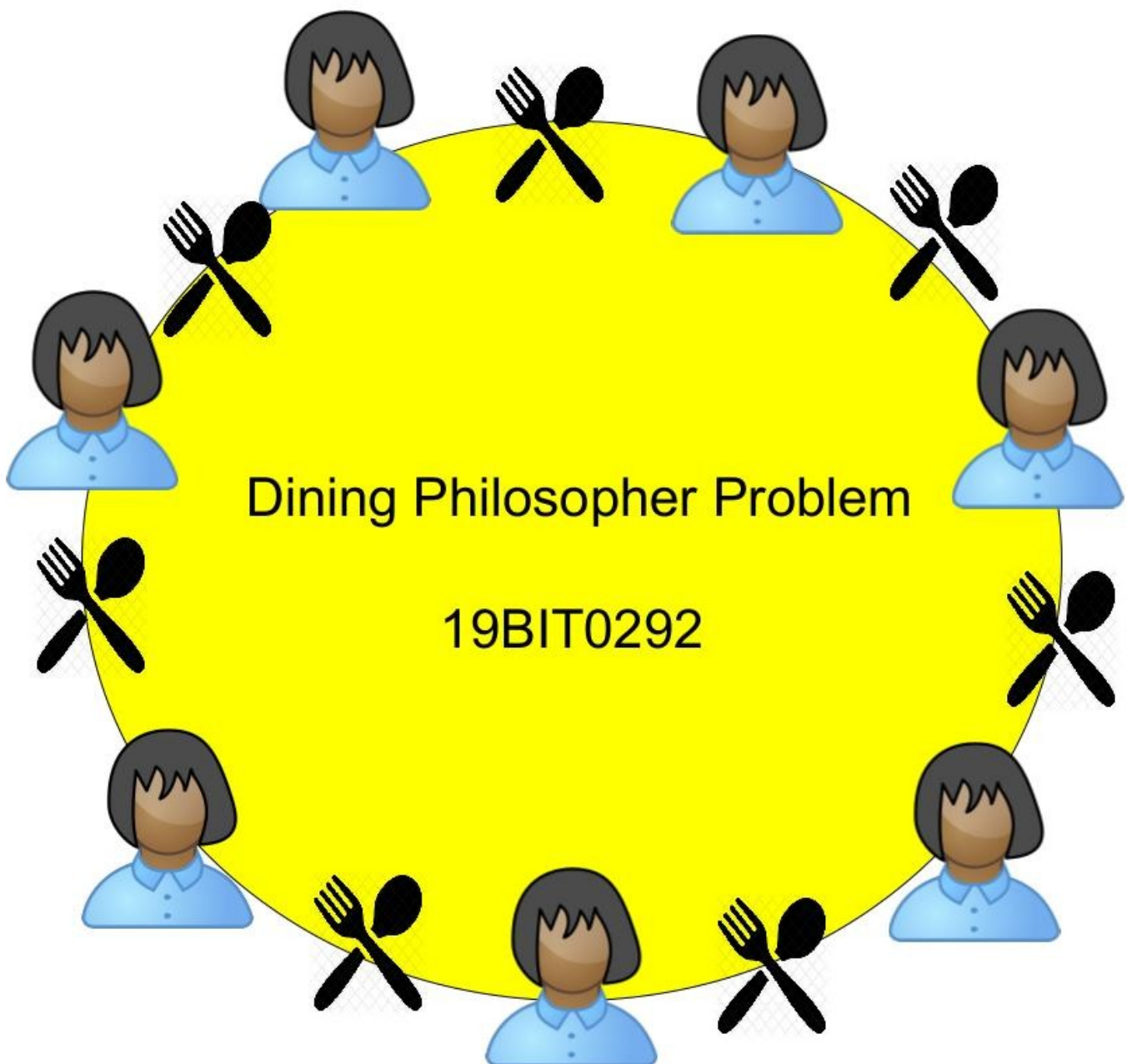
# OUTPUT

```
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\OneDrive\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\OneDrive\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\OneDrive\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;
```

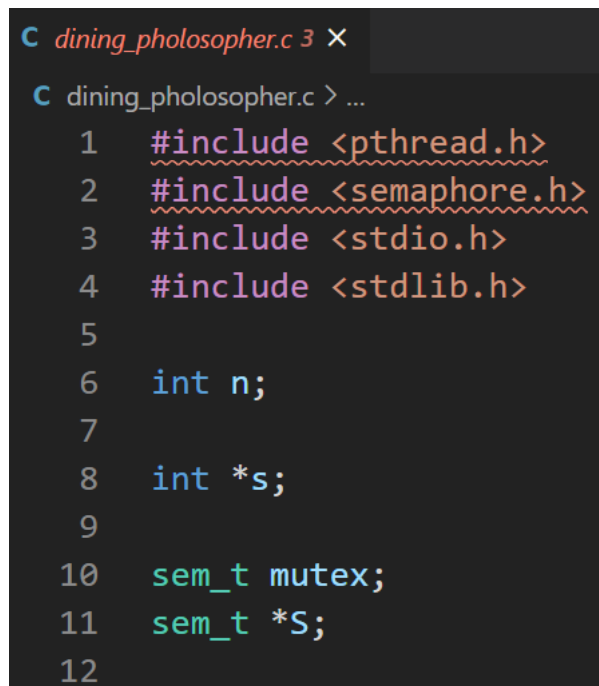
```
    sem_init(&mutex, 0, 1);
```

```
    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



The screenshot shows a code editor with a dark background. The title bar at the top reads "C dining\_pholosopher.c 3 X". The editor content shows the first 12 lines of a C program. Lines 1-4 are include statements for pthread.h, semaphore.h, stdio.h, and stdlib.h. Lines 5-9 are blank. Lines 6-8 declare variables: int n, int \*s, and sem\_t mutex. Lines 10-11 declare sem\_t \*S. Line 12 is blank.

```
C dining_pholosopher.c 3 X  
C dining_pholosopher.c > ...  
1  #include <pthread.h>  
2  #include <semaphore.h>  
3  #include <stdio.h>  
4  #include <stdlib.h>  
5  
6  int n;  
7  
8  int *s;  
9  
10 sem_t mutex;  
11 sem_t *S;  
12
```

```

13 void test(int phnum)
14 {
15     if (s[phnum] == 1 && s[(phnum + 4) % n] != 0 && s[(phnum + 1) % n] != 0)
16     {
17         s[phnum] = 0;
18
19         printf("Philosopher %d takes fork %d and %d\n",
20             phnum + 1, phnum + 1, (phnum + 4) % n + 1);
21
22         printf("Philosopher %d is eating\n", phnum + 1);
23
24         sem_post(&S[phnum]);
25     }
26 }
27
28 void take_fork(int phnum)
29 {
30
31     sem_wait(&mutex);
32
33     s[phnum] = 1;
34
35     test(phnum);
36
37     sem_post(&mutex);
38
39     sem_wait(&S[phnum]);
40 }
41
42 void put_fork(int phnum)
43 {
44     sem_wait(&mutex);
45
46     s[phnum] = 2;
47
48     printf("Philosopher %d putting fork %d and %d down\n",
49         phnum + 1, (phnum + 4) % n + 1, phnum + 1);
50     printf("Philosopher %d has finished eating and now thinking again\n", phnum + 1);
51
52     test((phnum + 4) % n);
53     test((phnum + 1) % n);
54
55     sem_post(&mutex);
56 }
57
58 void *philosopher(void *num)
59 {
60
61     int *i = num;
62
63     take_fork(*i);

```

```

63     take_fork(*i);
64
65     put_fork(*i);
66 }
67
68 int main()
69 {
70
71     int i, *p;
72     printf("(19BIT0292)Enter the total number of philosophers: ");
73     scanf("%d", &n);
74     pthread_t thread_id[n];
75     S = malloc(sizeof(sem_t) * n);
76     s = malloc(sizeof(int) * n);
77     for (i = 0; i < n; i++)
78         p[i] = i;
79     sem_init(&mutex, 0, 1);
80
81     for (i = 0; i < n; i++)
82     {
83         sem_init(&S[i], 0, 0);
84     }
85     for (i = 0; i < n; i++)
86     {
87         p = malloc(sizeof(int));
88         *p = i + 1;
89         pthread_create(&thread_id[i], NULL,
90             philosopher, p);
91     }
92     printf("Philosopher %d is thinking\n", i + 1);
93 }
94
95 for (i = 0; i < n; i++)
96     pthread_join(thread_id[i], NULL);
97 }

```

# 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\OneDrive\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\OneDrive\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 readerwriter.c 3 X
C readerwriter.c > wr(void *)
1  #include <pthread.h>
2  #include <semaphore.h>
3  #include <stdio.h>
4  #include <stdlib.h>
5
6  sem_t wrt;
7  pthread_mutex_t mutex;
8  int cnt = 1;
9  int numreader = 0;
10
11 void *wr(void *wno)
12 {
13     sem_wait(&wrt);
14     cnt = cnt * 2;
15     printf("Writer %d modified count to %d\n", (*((int *)wno)), cnt);
16     sem_post(&wrt);
17 }
18 void *rd(void *rno)
19 {
20     pthread_mutex_lock(&mutex);
21     numreader++;
22     if (numreader == 1)
23     {
24         sem_wait(&wrt);
25     }
26     pthread_mutex_unlock(&mutex);
27     printf("Reader %d: read count as %d\n", (*((int *)rno)), cnt);
28     pthread_mutex_lock(&mutex);
29     numreader--;
30     if (numreader == 0)
31     {
32         sem_post(&wrt);
33     }
34     pthread_mutex_unlock(&mutex);
35 }
```

```
36
37 void main()
38 {
39     pthread_t *r, *w;
40     int rn, wn, *n;
41     pthread_mutex_init(&mutex, NULL);
42     sem_init(&wrt, 0, 1);
43     printf("Enter the number of readers: ");
44     scanf("%d", &rn);
45     r = (pthread_t *)malloc(sizeof(pthread_t) * rn);
46     printf("Enter the number of writers: ");
47     scanf("%d", &wn);
48     w = (pthread_t *)malloc(sizeof(pthread_t) * wn);
49     for (int i = 0; i < rn; i++)
50     {
51         n = malloc(sizeof(int));
52         *n = i + 1;
53         pthread_create(&r[i], NULL, (void *)rd, (void *)n);
54     }
55     for (int i = 0; i < wn; i++)
56     {
57         n = malloc(sizeof(int));
58         *n = i + 1;
59         pthread_create(&w[i], NULL, (void *)wr, (void *)n);
60     }
61
62     for (int i = 0; i < rn; i++)
63         pthread_join(r[i], NULL);
64     for (int i = 0; i < wn; i++)
65         pthread_join(w[i], NULL);
66
67     pthread_mutex_destroy(&mutex);
68     sem_destroy(&wrt);
69 }
```

# 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\Desktop\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\Desktop\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\Desktop\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 upto now(in sequence): ");
for (i = 0; i < r; i++)
{
    printf("%d ", s[i]);
}

for (i = 0; i < r; i++)
{
    printf("\n(19BIT0292)Enter the remaining instance of resources number %d: ", i + 1);
    scanf("%d", &re[i]);
}

printf("\n(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[i]);
    printf(">");
}
}

```



# SCREENSHOT OF CODE

C banker.c 2 ×

C banker.c > main()

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  int main()
4  {
5      int n, r, i, j, f, fl, k = -1, *re, **p, **mx, **ne, *s, *jk;
6      printf("(19BIT0292)Enter the number of processes: ");
7      scanf("%d", &n);
8      p = (int **)malloc(sizeof(int *) * n);
9      mx = (int **)malloc(sizeof(int *) * n);
10     ne = (int **)malloc(sizeof(int *) * n);
11     jk = (int *)malloc(sizeof(int) * n);
12     printf("(19BIT0292)Number of types of resources: ");
13     scanf("%d", &r);
14     re = (int *)malloc(sizeof(int) * r);
15     s = (int *)calloc(r, sizeof(int));
16     printf("\n(19BIT0292)Enter the process allocation matrix\n");
17     for (i = 0; i < n; i++)
18     {
19         p[i] = (int *)malloc(sizeof(int) * r);
20         printf("\nP%d: ", i + 1);
21         for (j = 0; j < r; j++)
22             scanf("%d", &p[i][j]);
23     }
24     printf("\n(19BIT0292)Enter the process maximum matrix\n");
25     for (i = 0; i < n; i++)
26     {
27         mx[i] = (int *)malloc(sizeof(int) * r);
28         ne[i] = (int *)malloc(sizeof(int) * r);
29         printf("\nP%d: ", i + 1);
30         for (j = 0; j < r; j++)
31         {
32             scanf("%d", &mx[i][j]);
33             ne[i][j] = mx[i][j] - p[i][j];
34         }
35     }
36     printf("\n(19BIT0292)Need Matrix\n");
37     for (i = 0; i < n; i++)
38     {
39         printf("\nP%d: ", i + 1);
40         for (j = 0; j < r; j++)
41         {
42             printf("%d ", ne[i][j]);
43             s[j] += ne[i][j];
44         }
45     }
```

```

46 printf("\n(19BIT0292)Total number of resources used upto now(in sequence): ");
47 for (i = 0; i < r; i++)
48 {
49     printf("%d ", s[i]);
50 }
51
52 for (i = 0; i < r; i++)
53 {
54     printf("\n(19BIT0292)Enter the remaining instance of resources number %d: ", i + 1);
55     scanf("%d", re + i);
56 }
57 printf("(19BIT0292)Sequence: \n");
58 while (1)
59 {
60     fl = 0;
61     for (i = 0; i < n; i++)
62     {
63         if (p[i] == NULL)
64             continue;
65         else
66         {
67             f = 0;
68             for (j = 0; j < r; j++)
69             {
70                 if (ne[i][j] > re[j])
71                 {
72                     f = 1;
73                     break;
74                 }
75             }
76             if (f == 0)
77             {
78                 printf("P%d(", i + 1);
79                 for (j = 0; j < r - 1; j++)
80                 {
81                     re[j] += p[i][j];
82                     printf("%d,", re[j]);
83                 }
84                 re[r - 1] += p[i][r - 1];
85                 printf("%d)\n", re[r - 1]);
86                 p[i] = NULL;
87                 fl = 1;
88                 jk[++k] = i + 1;
89                 continue;
90             }
91         }
92     }
93     if (fl == 0)
94         break;
95 }
96
97 if (k == -1)
98     printf("\n\n(19BIT0292)Unsafe State");
99 else
100 {
101     printf("\n(19BIT0292)Safe State sequence <P%d", jk[0]);
102     for (i = 1; i < k + 1; i++)
103         printf(", P%d", jk[i]);
104     printf(">");
105 }

```

# 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 upto now(in sequence): 18 10 7
(19BIT0292)Enter the remaining instance of resources number 1: 0

(19BIT0292)Enter the remaining instance of resources number 2: 0

(19BIT0292)Enter the remaining instance of resources 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 upto now(in sequence): 1
8 10 7
(19BIT0292)Enter the remaining instance of resources number 1: 3      8 10 7

(19BIT0292)Enter the remaining instance of resources number 2: 3

(19BIT0292)Enter the remaining instance of resources 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](#)**