**Experiment No: 06**

**Aim:** Write a program to implement solution of Producer consumer problem through Semaphore.

**Theory:** The producer-consumer problem is an example of a [multi-process synchronization](https://www.geeksforgeeks.org/introduction-of-process-synchronization/) problem. The problem describes two processes, the producer and the consumer that shares a common fixed-size buffer use it as a [queue](https://www.geeksforgeeks.org/queue-data-structure/).

* The producer’s job is to generate data, put it into the buffer, and start again.
* At the same time, the consumer is consuming the data (i.e., removing it from the buffer), one piece at a time.

**Problem:** Given the common fixed-size buffer, the task is to make sure that the producer can’t add data into the buffer when it is full and the consumer can’t remove data from an empty buffer.

**Solution:** The producer is to either go to sleep or discard data if the buffer is full. The next time the consumer removes an item from the buffer, it notifies the producer, who starts to fill the buffer again. In the same manner, the consumer can go to sleep if it finds the buffer to be empty. The next time the producer puts data into the buffer, it wakes up the sleeping consumer.

**Program:**

#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\n2.Consumer\n3.Exit");

while(1)

{

printf("\nEnter 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("\nProducer produces the item %d",x);

mutex=signal(mutex);

}

void consumer()

{

mutex=wait(mutex);

full=wait(full);

empty=signal(empty);

printf("\nConsumer consumes item %d",x);

x--;

mutex=signal(mutex);

}

**Output:**

1.Producer

2.Consumer

3.Exit

Enter your choice:1

Producer produces the item 1

Enter your choice:1

Producer produces the item 2

Enter your choice:1

Producer produces the 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:1

Producer produces the item 1

Enter your choice:2

Consumer consumes item 1

Enter your choice:1

Producer produces the item 1

Enter your choice:1

Producer produces the item 2

Enter your choice:1

Producer produces the item 3

Enter your choice:1

Buffer is full!!

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:3

**Conclusion:** In this experiment we have implemented Producer consumer problem through Semaphore.