OS:Experiment No. 6

Name: Ayush Jain
SAP ID: 60004200132
Batch: B2
Computer Engineering

Aim- There is a service counter which has a limited waiting queue outside it. It works as follows:

- The counter remains open till the waiting queue is not empty
- If the queue is already full, the new customer simply leaves
- If the queue becomes empty, the outlet doors will be closed (service personnel sleep)
- Whenever a customer arrives at the closed outlet, he/she needs to wake the person at the counter with a wake-up call

Implement the above-described problem using semaphores or mutuexes along with threads. Also show how it works, if there are 2 service personnel, and a single queue. Try to simulate all possible events that can take place, in the above scenario.

Problem Statement-

1) Use Producer Consumer Concept to implement above scenario which treats the queue as buffer.
2) Use Semaphore and mutex for concurrency control and queue status update(full/empty)
3) Use multithreading for implementation

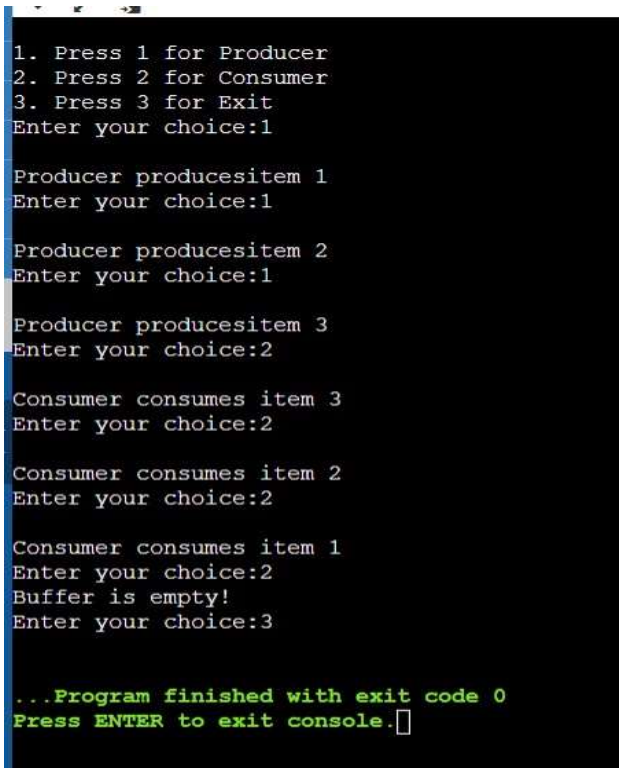
Code-

```
#include <stdio.h> #include
<stdlib.h>
int mutex = 1; int
full = 0; int empty =
5, x = 0;
int wait (int n)
{     n--;
return n;
}
int Signal(int n)
{
n++;
   return n;
}
void producer()
{
```

```
    mutex =wait(mutex);
empty = wait(empty);
x=Signal(x);
printf("\nProducer produces"
"item %d",
          x);
full=Signal(full);
    mutex =Signal(mutex);
}
void consumer()
{
    mutex=wait(mutex);
    full=wait(full);

    printf("\nConsumer consumes "
"item %d",            x);
    x=wait(x);
empty=Signal(empty);
mutex=Signal(mutex);
}

int main()
{
    int n, i;
                    printf("\n1. Press 1 for
  Producer"            "\n2. Press 2 for
                                 Consumer"
          "\n3. Press 3 for Exit");
#pragma omp critical

    for (i = 1; i > 0; i++) {

        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;
      }
   }
}
```

Output:



Conclusion:

Solving the producer consumer problem via use of semaphore and mutex variables.