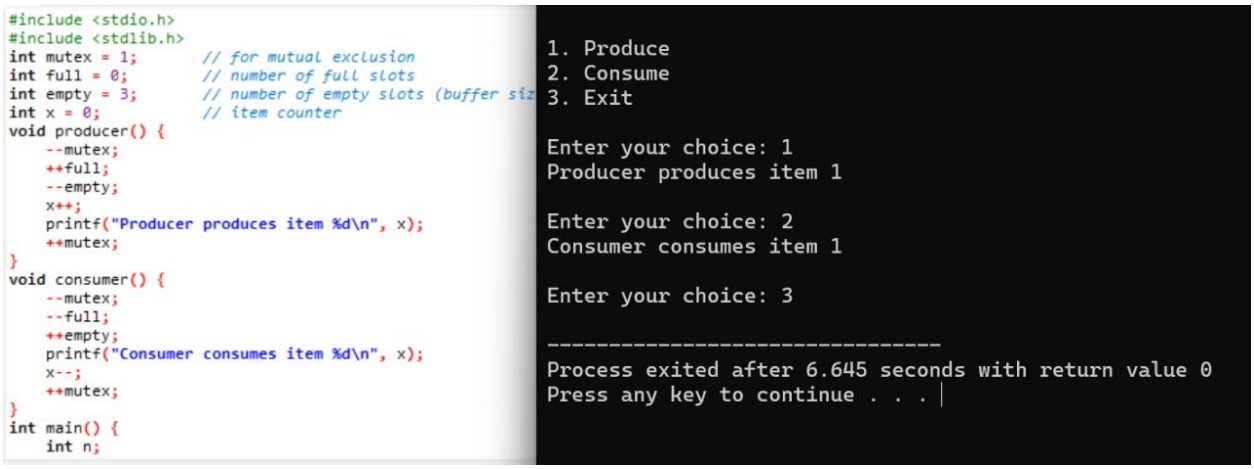
18 Construct a C program to simulate producer-consumer problem using semaphores.

```
#include <stdio.h>
#include <stdlib.h>
int mutex = 1;       // for mutual exclusion
int full = 0;        // number of full slots
int empty = 3;       // number of empty slots (buffer size)
int x = 0;           // item counter
void producer() {
   --mutex;
   ++full;
   --empty;
   x++;
   printf("Producer produces item %d\n", x);
   ++mutex;
}
void consumer() {
   --mutex;
   --full;
   ++empty;
   printf("Consumer consumes item %d\n", x);
   x--;
   ++mutex;
}
int main() {
   int n;
   printf("\n1. Produce\n2. Consume\n3. Exit\n");
```

```
    while (1) {
        printf("\nEnter your choice: ");
        scanf("%d", &n);
        switch (n) {
            case 1:
                if ((mutex == 1) && (empty != 0))
                    producer();
                else
                    printf("Buffer is full!\n");
                break;
            case 2:
                if ((mutex == 1) && (full != 0))
                    consumer();
                else
                    printf("Buffer is empty!\n");
                break;
            case 3:
                exit(0);
                break;
            default:
                printf("Invalid choice!\n");
        }
    }
    return 0;
}
```

OUTPUT-



```c
#include <stdio.h>
#include <stdlib.h>
int mutex = 1;       // for mutual exclusion
int full = 0;        // number of full slots
int empty = 3;       // number of empty slots (buffer siz
int x = 0;           // item counter
void producer() {
    --mutex;
    ++full;
    --empty;
    x++;
    printf("Producer produces item %d\n", x);
    ++mutex;
}
void consumer() {
    --mutex;
    --full;
    ++empty;
    printf("Consumer consumes item %d\n", x);
    x--;
    ++mutex;
}
int main() {
    int n;
```

```
1. Produce
2. Consume
3. Exit

Enter your choice: 1
Producer produces item 1

Enter your choice: 2
Consumer consumes item 1

Enter your choice: 3

--------------------------------
Process exited after 6.645 seconds with return value 0
Press any key to continue . . .
```