Program-4

Aim: Write a program to simulate producer-consumer problem using semaphores.

Source code:

```
#include <stdio.h>
#include <stdlib.h>
// Global variables
int mutex = 1;   // Mutex for synchronization
int full = 0;    // Counter for items in the buffer
int empty = 3;   // Counter for available space in the buffer
int x = 0;       // Item counter
// Function prototypes
int wait(int);
int signal(int);
void producer();
void consumer();
int main() {
   int n;
   printf("\n1. Producer\n2. Consumer\n3. Exit");
   // Main loop to repeatedly display the menu and wait for user choice
   while (1) {
      printf("\nEnter your choice: ");
      scanf("%d", &n);
      switch (n) {
         case 1:
            if ((mutex == 1) && (empty != 0)) {  // Check if mutex is available and there is space in the buffer
               producer();
            } else {
               printf("Buffer is full!!");
            }
            break;
         case 2:
            if ((mutex == 1) && (full != 0)) {  // Check if mutex is available and there are items to consume
```

```
                consumer();
            } else {
                printf("Buffer is empty!!");
            }
            break;
        case 3:
            exit(0);  // Exit the program
            break;
        default:
            printf("Invalid choice. Please enter 1, 2, or 3.");
      }
   }
   return 0;
}
// Wait function (decrements a value and returns it)
int wait(int s) {
   return (--s);
}
// Signal function (increments a value and returns it)
int signal(int s) {
   return (++s);
}
// Producer function
void producer() {
   mutex = wait(mutex); // Acquire mutex lock
   full = signal(full); // Increment the full counter (producing an item)
   empty = wait(empty); // Decrement the empty counter (space is reduced)
   x++;                 // Increment the item counter
   printf("\nProducer produces item %d", x);
   mutex = signal(mutex); // Release mutex lock
}
// Consumer function
void consumer() {
```

```
    mutex = wait(mutex); // Acquire mutex lock
    full = wait(full);   // Decrement the full counter (consuming an item)
    empty = signal(empty); // Increment the empty counter (space is increased)
    printf("\nConsumer consumes item %d", x);
    x--;                 // Decrement the item counter
    mutex = signal(mutex); // Release mutex lock
}
```

Sample output:

1. Producer

2. Consumer

3. Exit

Enter your choice: 1

Producer produces item 1

Enter your choice: 1

Producer produces item 2

Enter your choice: 1

Producer produces 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: 3