

Department of Computer Science and Engineering

S.G.Shivanirudh , 185001146, Semester IV

8 March 2020

UCS1411 - Operating Systems Laboratory

Lab Exercise 6: Implementation of Producer/Consumer Problem using Semaphores

Objective:

Develop the following applications that uses semaphores concepts using to implement producer consumer problem.

Code:

Q1.To write a C program to create parent/child processes to implement the producer/consumer problem using semaphores in pthread library.

```
1
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <string.h>
5 #include <semaphore.h>
```

```

6 #include <pthread.h> // for semaphore operations sem_init,
   sem_wait,sem_post
7 #include <sys/ipc.h>
8 #include <sys/shm.h>
9 #include <sys/sem.h>
10 #include <sys/wait.h>
11 #include <sys/errno.h>
12 #include <sys/types.h>
13 #include<unistd.h>
14
15 extern int  errno;
16
17 //Size of the shared buffer
18 #define SIZE 10
19 //Size of shared variable=1 byte
20 #define VARSIZE 1
21 //Maximum input limit
22 #define INPUTSIZE 20
23 //shared memory permissions
24 #define SHMPERM 0666
25
26 void my_wait(int *sem){
27     while(*sem<=0);
28     (*sem)--;
29 }
30
31 void my_signal(int *sem){
32     (*sem)++;
33 }
34
35 // Producer function
36 void produce(char *input_string,int *empty,int *full,int *
   mutex,int *pctr,char *buff,int *cctr){
37     int i=0;
38     while (1){
39
40         if(i>=strlen(input_string)){
41             printf("\n Producer %d exited \n",getpid());
42             wait(NULL);
43             exit(1);
44         }
45
46         //Acquire semaphore empty
47         printf("\nProducer %d trying to aquire Semaphore
   Empty \n",getpid());

```

```

48         my_wait(empty);
49         printf("\nProducer %d successfully aquired Semaphore
Empty \n",getpid());
50
51         //Acquire semaphore mutex
52         printf("\nProducer %d trying to aquire Semaphore
Mutex \n",getpid());
53         my_wait(mutex);
54         printf("\nProducer %d successfully aquired Semaphore
Mutex \n",getpid());
55
56         //Critical Section
57         buff[*pctr]=input_string[i];
58
59         printf("\nProducer %d Produced Item [ %c ] \n",getpid
(),input_string[i]);
60         i++;
61
62         (*pctr)++;
63         printf("\nItems in Buffer %d \n",*pctr-*cctr);
64
65         //Release semaphore mutex
66         my_signal(mutex);
67         printf("\nProducer %d released Semaphore Mutex \n",
getpid());
68
69         //Release semaphore full
70         my_signal(full);
71         printf("\nProducer %d released Semaphore Full \n",
getpid());
72
73         //Remainder section
74         sleep(2/random());
75     }
76 }
77
78 // Consumer function
79 void consume(char *input_string,int *empty,int *full,int *
mutex,int *cctr,char *buff,int *pctr){
80     int i=0;
81     while (1){
82         if(i>=strlen(input_string)){
83             printf("\n Consumer %d exited \n",getpid());
84             exit(1);
85         }

```

```

86
87     //Acquire semaphore full
88     printf("\nConsumer %d trying to aquire Semaphore Full
\n",getpid());
89     my_wait(full);
90     printf("\nConsumer %d successfully aquired Semaphore
Full \n",getpid());
91
92     //Acquire semaphore mutex
93     printf("\nConsumer %d trying to aquire Semaphore
Mutex \n",getpid());
94     my_wait(mutex);
95     printf("\nConsumer %d successfully aquired Semaphore
Mutex\n",getpid());
96
97     //Critical Section
98     printf("\nConsumer %d Consumed Item [ %c ] \n",getpid
(),buff[*cctr]);
99     buff[*cctr]=' ';
100     (*cctr)++;
101
102     printf("\nItems in Buffer %ld \n",strlen(buff)-*cctr)
;
103     i++;
104
105     //Release semaphore mutex
106     my_signal(mutex);
107     printf("\nConsumer %d released Semaphore Mutex \n",
getpid());
108
109     //Release semaphore empty
110     my_signal(empty);
111     printf("\nConsumer %d released Semaphore Empty \n",
getpid());
112
113     //Remainder Section
114     sleep(1);
115 }
116 }
117
118 int main(){
119     //ID for shared memory bufer
120     int shmid;
121
122     /*

```

```

123
124     IMPORTANT:
125     The semaphores, integer pointers in this case are all
126     initialised to NULL for now, instead of the respective
127     values they should be taking.
128     Initialisation is done later. Refer further notes.
129
130     */
131
132     //Semaphore empty
133     int empty_id;
134     int *empty=(int*)malloc(sizeof(int));
135
136     //Semaphore full
137     int full_id;
138     int *full=(int*)malloc(sizeof(int));
139
140     //Semaphore mutex
141     int mutex_id;
142     int *mutex=(int*)malloc(sizeof(int));
143
144     //Buffer to read from/write onto the shared memory
145     char *buff;
146
147     //Input string
148     char *input_string=(char*)malloc(20*sizeof(char));
149
150     //Buffer counters in producer and consumer respectively
151     int pctr=0,cctr=0;
152
153     pid_t temp_pid;
154
155     //Acquiring memory for shared memory
156     shm_id = shmget(IPC_PRIVATE, SIZE, IPC_CREAT | IPC_EXCL
157     | SHMPERM );
158     full_id = shmget(IPC_PRIVATE, SIZE, IPC_CREAT | IPC_EXCL
159     | SHMPERM);
160     empty_id= shmget(IPC_PRIVATE, SIZE, IPC_CREAT | IPC_EXCL
161     | SHMPERM);
162     mutex_id= shmget(IPC_PRIVATE, SIZE, IPC_CREAT | IPC_EXCL
163     | SHMPERM);
164
165     //Attaching buffer to memory location
166     buff = shmat(shm_id,(char*)0,0);
167     full = shmat(full_id,(int*)0,0);

```

```

162     empty = shmat(empty_id,(int*)0,0);
163     mutex =shmat(mutex_id,(int*)0,0);
164
165     //Initialising the semaphores
166     /*
167
168     IMPORTANT:
169     Note that the initialisation is done after the attachment
170     to the memory locations.
171     Doing these two operations in reverse, for some yet
172     unknown reason, leads to all the semaphores being
173     initialised to 0.
174
175     */
176
177     *empty=SIZE;
178     *full=0;
179     *mutex=1;
180
181     printf("\n Main Process \n");
182     printf("\nEnter string: ");scanf(" %s",input_string);
183     printf("Entered string : %s",input_string);
184
185     temp_pid=fork();
186     if(temp_pid>0){
187         produce(input_string,empty,full,mutex,&pctr,buff,&
188 cctr);
189     }
190     else{
191         consume(input_string,empty,full,mutex,&cctr,buff,&
192 pctr);
193     }
194
195     //Detaching buffer from memory location
196     shmdt(buff);
197     shmdt(mutex);
198     shmdt(empty);
199     shmdt(full);
200
201     //Destroying acquired location
202     shmctl(shmid, IPC_RMID, NULL);
203     shmctl(mutex_id, IPC_RMID, NULL);
204     shmctl(empty_id, IPC_RMID, NULL);
205     shmctl(full_id, IPC_RMID, NULL);

```

```

202
203     printf("\n Main process exited \n\n");
204     return(0);
205 }

```

Output:

```

1
2  Main Process
3
4 Enter string: asdfgh
5 Entered string : asdfgh
6 Producer 3465 trying to aquire Semaphore Empty
7
8 Producer 3465 successfully aquired Semaphore Empty
9
10 Producer 3465 trying to aquire Semaphore Mutex
11
12 Producer 3465 successfully aquired Semaphore Mutex
13
14 Producer 3465 Produced Item [ a ]
15
16 Items in Buffer 1
17 Entered string : asdfgh
18
19 Producer 3465 released Semaphore Mutex
20 Consumer 3466 trying to aquire Semaphore Full
21
22 Producer 3465 released Semaphore Full
23
24 Consumer 3466 successfully aquired Semaphore Full
25
26 Consumer 3466 trying to aquire Semaphore Mutex
27
28 Consumer 3466 successfully aquired Semaphore Mutex
29
30 Consumer 3466 Consumed Item [ a ]
31
32
33 Producer 3465 trying to aquire Semaphore Empty
34 Items in Buffer 0
35
36
37 Producer 3465 successfully aquired Semaphore Empty
38 Consumer 3466 released Semaphore Mutex

```

39
40 Producer 3465 trying to aquire Semaphore Mutex
41
42
43 Consumer 3466 released Semaphore Empty
44 Producer 3465 successfully aquired Semaphore Mutex
45
46 Producer 3465 Produced Item [s]
47
48 Items in Buffer 2
49
50 Producer 3465 released Semaphore Mutex
51
52 Producer 3465 released Semaphore Full
53
54 Producer 3465 trying to aquire Semaphore Empty
55
56 Producer 3465 successfully aquired Semaphore Empty
57
58 Producer 3465 trying to aquire Semaphore Mutex
59
60 Producer 3465 successfully aquired Semaphore Mutex
61
62 Producer 3465 Produced Item [d]
63
64 Items in Buffer 3
65
66 Producer 3465 released Semaphore Mutex
67
68 Producer 3465 released Semaphore Full
69
70 Producer 3465 trying to aquire Semaphore Empty
71
72 Producer 3465 successfully aquired Semaphore Empty
73
74 Producer 3465 trying to aquire Semaphore Mutex
75
76 Producer 3465 successfully aquired Semaphore Mutex
77
78 Producer 3465 Produced Item [f]
79
80 Items in Buffer 4
81
82 Producer 3465 released Semaphore Mutex
83

84 Producer 3465 released Semaphore Full
85
86 Producer 3465 trying to aquire Semaphore Empty
87
88 Producer 3465 successfully aquired Semaphore Empty
89
90 Producer 3465 trying to aquire Semaphore Mutex
91
92 Producer 3465 successfully aquired Semaphore Mutex
93
94 Producer 3465 Produced Item [g]
95
96 Items in Buffer 5
97
98 Producer 3465 released Semaphore Mutex
99
100 Producer 3465 released Semaphore Full
101
102 Producer 3465 trying to aquire Semaphore Empty
103
104 Producer 3465 successfully aquired Semaphore Empty
105
106 Producer 3465 trying to aquire Semaphore Mutex
107
108 Producer 3465 successfully aquired Semaphore Mutex
109
110 Producer 3465 Produced Item [h]
111
112 Items in Buffer 6
113
114 Producer 3465 released Semaphore Mutex
115
116 Producer 3465 released Semaphore Full
117
118 Producer 3465 exited
119
120 Consumer 3466 trying to aquire Semaphore Full
121
122 Consumer 3466 successfully aquired Semaphore Full
123
124 Consumer 3466 trying to aquire Semaphore Mutex
125
126 Consumer 3466 successfully aquired Semaphore Mutex
127
128 Consumer 3466 Consumed Item [s]

129
130 Items in Buffer 4
131
132 Consumer 3466 released Semaphore Mutex
133
134 Consumer 3466 released Semaphore Empty
135
136 Consumer 3466 trying to aquire Semaphore Full
137
138 Consumer 3466 successfully aquired Semaphore Full
139
140 Consumer 3466 trying to aquire Semaphore Mutex
141
142 Consumer 3466 successfully aquired Semaphore Mutex
143
144 Consumer 3466 Consumed Item [d]
145
146 Items in Buffer 3
147
148 Consumer 3466 released Semaphore Mutex
149
150 Consumer 3466 released Semaphore Empty
151
152 Consumer 3466 trying to aquire Semaphore Full
153
154 Consumer 3466 successfully aquired Semaphore Full
155
156 Consumer 3466 trying to aquire Semaphore Mutex
157
158 Consumer 3466 successfully aquired Semaphore Mutex
159
160 Consumer 3466 Consumed Item [f]
161
162 Items in Buffer 2
163
164 Consumer 3466 released Semaphore Mutex
165
166 Consumer 3466 released Semaphore Empty
167
168 Consumer 3466 trying to aquire Semaphore Full
169
170 Consumer 3466 successfully aquired Semaphore Full
171
172 Consumer 3466 trying to aquire Semaphore Mutex
173

```

174 Consumer 3466 successfully aquired Semaphore Mutex
175
176 Consumer 3466 Consumed Item [ g ]
177
178 Items in Buffer 1
179
180 Consumer 3466 released Semaphore Mutex
181
182 Consumer 3466 released Semaphore Empty
183
184 Consumer 3466 trying to aquire Semaphore Full
185
186 Consumer 3466 successfully aquired Semaphore Full
187
188 Consumer 3466 trying to aquire Semaphore Mutex
189
190 Consumer 3466 successfully aquired Semaphore Mutex
191
192 Consumer 3466 Consumed Item [ h ]
193
194 Items in Buffer 0
195
196 Consumer 3466 released Semaphore Mutex
197
198 Consumer 3466 released Semaphore Empty

```

Q2.Modify the program as separate client / server process programs to generate 'N' random numbers in producer and write them into shared memory. Consumer process should read them from shared memory and display them in terminal

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 // For semaphore operations sem_init,sem_wait,sem_post
5 #include <semaphore.h>
6 #include <pthread.h>
7 #include <unistd.h>
8 #include <sys/ipc.h>
9 #include <sys/shm.h>
10 #include <sys/sem.h>
11 #include <sys/wait.h>
12 #include <sys/errno.h>

```

```

13 #include <sys/types.h>
14 #include<unistd.h>
15 extern int errno;
16 #define SIZE 10 /* size of the shared buffer */
17 #define VARSIZE 1 /* size of shared variable = 1 byte */
18 #define INPUTSIZE 20
19 #define SHMPERM 0666 /* shared memory permissions */
20 int segid; /* ID for shared memory buffer */
21 int empty_id;
22 int full_id;
23 int mutex_id;
24 char *buff;
25 char *input_string;
26 sem_t *empty;
27 sem_t *full;
28 sem_t *mutex;
29 int p = 0;
30 int main()
31 {
32     int i = 0;
33     pid_t temp_pid;
34     segid = shmget(104, SIZE, IPC_CREAT | IPC_EXCL | SHMPERM
35 );
36     empty_id=shmget(105, sizeof(sem_t), IPC_CREAT | IPC_EXCL
37 | SHMPERM);
38     full_id=shmget(106, sizeof(sem_t), IPC_CREAT | IPC_EXCL |
39 SHMPERM);
40     mutex_id=shmget(107, sizeof(sem_t), IPC_CREAT | IPC_EXCL |
41 SHMPERM);
42     buff = shmat(segid, (char *)0, 0);
43     empty = shmat(empty_id, (char *)0, 0);
44     full = shmat(full_id, (char *)0, 0);
45     mutex = shmat(mutex_id, (char *)0, 0);
46     // Initializing Semaphores Empty, Full & Mutex
47     sem_init(empty, 1, 10);
48     sem_init(full, 1, 0);
49     sem_init(mutex, 1, 1);
50     printf("\nProducer Process Started\n");
51     while (i < 10)
52     {
53         int val = random()%10;
54         printf("\nProducer %d trying to acquire Semaphore
55 Empty\n", getpid());
56         sem_wait(empty);
57         printf("\nProducer %d successfully acquired Semaphore

```

```

    Empty\n", getpid());
53     printf("\nProducer %d trying to acquire Semaphore
Mutex\n", getpid());
54     sem_wait(mutex);
55     printf("\nProducer %d successfully acquired Semaphore
Mutex\n", getpid());
56     buff[p] = (char)(val + 48);
57     printf("\nProducer %d Produced Item [%d]\n", getpid()
, val);
58     i++;
59     p++;
60     printf("\nItems produced: %d\n", p);
61     sem_post(mutex);
62     printf("\nProducer %d released Semaphore Mutex\n",
getpid());
63     sem_post(full);
64     printf("\nProducer %d released Semaphore Full\n",
getpid());
65     sleep(2);
66 }
67 shmdt(buff);
68 shmdt(empty);
69 shmdt(full);
70 shmdt(mutex);
71 printf("\nProducer Process Ended\n");
72 return(0);
73 }

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 //For semaphore operations - sem_init, sem_wait, sem_post
5 #include <semaphore.h>
6 #include <pthread.h>
7 #include <unistd.h>
8 #include <sys/ipc.h>
9 #include <sys/shm.h>
10 #include <sys/sem.h>
11 #include <sys/wait.h>
12 #include <sys/errno.h>
13 #include <sys/types.h>
14 #include <unistd.h>
15 extern int errno;
16
17 #define SIZE 10 /* size of the shared buffer */
18 #define VARSIZE 1 /* size of shared variable = 1 byte */

```

```

19 #define INPUTSIZE 20
20 #define SHMPERM 0666 /* shared memory permissions */
21
22 int segid; /* ID for shared memory buffer */
23 int empty_id;
24 int full_id;
25 int mutex_id;
26
27 char *buff;
28 char *input_string;
29 sem_t *empty;
30 sem_t *full;
31 sem_t *mutex;
32 int p = 0, c = 0;
33 int main()
34 {
35     int i = 0;
36     pid_t temp_pid;
37     segid = shmget (104, SIZE, IPC_EXCL | SHMPERM );
38     empty_id = shmget(105, sizeof(sem_t), IPC_EXCL | SHMPERM)
39     ;
40     full_id = shmget(106, sizeof(sem_t), IPC_EXCL | SHMPERM);
41     mutex_id=shmget(107, sizeof(sem_t), IPC_EXCL | SHMPERM);
42     buff = shmat(segid, (char *)0, 0);
43     empty = shmat(empty_id, (char *)0, 0);
44     full = shmat(full_id, (char *)0, 0);
45     mutex = shmat(mutex_id, (char *)0, 0);
46     printf("\nConsumer Process Started\n");
47     while (i < 10)
48     {
49         printf("\nConsumer %d trying to acquire Semaphore
50 Full\n", getpid());
51         sem_wait(full);
52         printf("\nConsumer %d successfully acquired Semaphore
53 Full\n", getpid());
54         printf("\nConsumer %d trying to acquire Semaphore
55 Mutex\n", getpid());
56         sem_wait(mutex);
57         printf("\nConsumer %d successfully acquired Semaphore
58 Mutex\n", getpid());
59         printf("\nConsumer %d Consumed Item [%c]\n", getpid()
60 , buff[c]);
61         buff[c]=' ';
62         c++;
63         printf("\nItems consumed: %d\n", i+1);

```

```

58         i++;
59         sem_post(mutex);
60         printf("\nConsumer %d released Semaphore Mutex\n",
getpid());
61         sem_post(empty);
62         printf("\nConsumer %d released Semaphore Empty\n",
getpid());
63         sleep(1);
64     }
65     shmdt(buff);
66     shmdt(empty);
67     shmdt(full);
68     shmdt(mutex);
69     sleep(10);
70     shmctl(segid, IPC_RMID, NULL);
71     semctl(empty_id, 0, IPC_RMID, NULL);
72     semctl(full_id, 0, IPC_RMID, NULL);
73     semctl(mutex_id, 0, IPC_RMID, NULL);
74     sem_destroy(empty);
75     sem_destroy(full);
76     sem_destroy(mutex);
77     printf("\nConsumer Process Ended\n");
78     return(0);
79 }

```

Output:

```

1 Producer Process Started
2
3 Producer 3733 trying to acquire Semaphore Empty
4
5 Producer 3733 successfully acquired Semaphore Empty
6
7 Producer 3733 trying to acquire Semaphore Mutex
8
9 Producer 3733 successfully acquired Semaphore Mutex
10
11 Producer 3733 Produced Item [3]
12
13 Items produced: 1
14
15 Producer 3733 released Semaphore Mutex
16
17 Producer 3733 released Semaphore Full
18

```

19 Producer 3733 trying to acquire Semaphore Empty
20
21 Producer 3733 successfully acquired Semaphore Empty
22
23 Producer 3733 trying to acquire Semaphore Mutex
24
25 Producer 3733 successfully acquired Semaphore Mutex
26
27 Producer 3733 Produced Item [6]
28
29 Items produced: 2
30
31 Producer 3733 released Semaphore Mutex
32
33 Producer 3733 released Semaphore Full
34
35 Producer 3733 trying to acquire Semaphore Empty
36
37 Producer 3733 successfully acquired Semaphore Empty
38
39 Producer 3733 trying to acquire Semaphore Mutex
40
41 Producer 3733 successfully acquired Semaphore Mutex
42
43 Producer 3733 Produced Item [7]
44
45 Items produced: 3
46
47 Producer 3733 released Semaphore Mutex
48
49 Producer 3733 released Semaphore Full
50
51 Producer 3733 trying to acquire Semaphore Empty
52
53 Producer 3733 successfully acquired Semaphore Empty
54
55 Producer 3733 trying to acquire Semaphore Mutex
56
57 Producer 3733 successfully acquired Semaphore Mutex
58
59 Producer 3733 Produced Item [5]
60
61 Items produced: 4
62
63 Producer 3733 released Semaphore Mutex

64
65 Producer 3733 released Semaphore Full
66
67 Producer 3733 trying to acquire Semaphore Empty
68
69 Producer 3733 successfully acquired Semaphore Empty
70
71 Producer 3733 trying to acquire Semaphore Mutex
72
73 Producer 3733 successfully acquired Semaphore Mutex
74
75 Producer 3733 Produced Item [3]
76
77 Items produced: 5
78
79 Producer 3733 released Semaphore Mutex
80
81 Producer 3733 released Semaphore Full
82
83 Producer 3733 trying to acquire Semaphore Empty
84
85 Producer 3733 successfully acquired Semaphore Empty
86
87 Producer 3733 trying to acquire Semaphore Mutex
88
89 Producer 3733 successfully acquired Semaphore Mutex
90
91 Producer 3733 Produced Item [5]
92
93 Items produced: 6
94
95 Producer 3733 released Semaphore Mutex
96
97 Producer 3733 released Semaphore Full
98
99 Producer 3733 trying to acquire Semaphore Empty
100
101 Producer 3733 successfully acquired Semaphore Empty
102
103 Producer 3733 trying to acquire Semaphore Mutex
104
105 Producer 3733 successfully acquired Semaphore Mutex
106
107 Producer 3733 Produced Item [6]
108

109 Items produced: 7
110
111 Producer 3733 released Semaphore Mutex
112
113 Producer 3733 released Semaphore Full
114
115 Producer 3733 trying to acquire Semaphore Empty
116
117 Producer 3733 successfully acquired Semaphore Empty
118
119 Producer 3733 trying to acquire Semaphore Mutex
120
121 Producer 3733 successfully acquired Semaphore Mutex
122
123 Producer 3733 Produced Item [2]
124
125 Items produced: 8
126
127 Producer 3733 released Semaphore Mutex
128
129 Producer 3733 released Semaphore Full
130
131 Producer 3733 trying to acquire Semaphore Empty
132
133 Producer 3733 successfully acquired Semaphore Empty
134
135 Producer 3733 trying to acquire Semaphore Mutex
136
137 Producer 3733 successfully acquired Semaphore Mutex
138
139 Producer 3733 Produced Item [9]
140
141 Items produced: 9
142
143 Producer 3733 released Semaphore Mutex
144
145 Producer 3733 released Semaphore Full
146
147 Producer 3733 trying to acquire Semaphore Empty
148
149 Producer 3733 successfully acquired Semaphore Empty
150
151 Producer 3733 trying to acquire Semaphore Mutex
152
153 Producer 3733 successfully acquired Semaphore Mutex

```
154
155 Producer 3733 Produced Item [1]
156
157 Items produced: 10
158
159 Producer 3733 released Semaphore Mutex
160
161 Producer 3733 released Semaphore Full
162
163 Producer Process Ended

  1 Consumer Process Started
  2
  3 Consumer 3734 trying to acquire Semaphore Full
  4
  5 Consumer 3734 successfully acquired Semaphore Full
  6
  7 Consumer 3734 trying to acquire Semaphore Mutex
  8
  9 Consumer 3734 successfully acquired Semaphore Mutex
 10
 11 Consumer 3734 Consumed Item [3]
 12
 13 Items consumed: 1
 14
 15 Consumer 3734 released Semaphore Mutex
 16
 17 Consumer 3734 released Semaphore Empty
 18
 19 Consumer 3734 trying to acquire Semaphore Full
 20
 21 Consumer 3734 successfully acquired Semaphore Full
 22
 23 Consumer 3734 trying to acquire Semaphore Mutex
 24
 25 Consumer 3734 successfully acquired Semaphore Mutex
 26
 27 Consumer 3734 Consumed Item [6]
 28
 29 Items consumed: 2
 30
 31 Consumer 3734 released Semaphore Mutex
 32
 33 Consumer 3734 released Semaphore Empty
 34
 35 Consumer 3734 trying to acquire Semaphore Full
```

36
37 Consumer 3734 successfully acquired Semaphore Full
38
39 Consumer 3734 trying to acquire Semaphore Mutex
40
41 Consumer 3734 successfully acquired Semaphore Mutex
42
43 Consumer 3734 Consumed Item [7]
44
45 Items consumed: 3
46
47 Consumer 3734 released Semaphore Mutex
48
49 Consumer 3734 released Semaphore Empty
50
51 Consumer 3734 trying to acquire Semaphore Full
52
53 Consumer 3734 successfully acquired Semaphore Full
54
55 Consumer 3734 trying to acquire Semaphore Mutex
56
57 Consumer 3734 successfully acquired Semaphore Mutex
58
59 Consumer 3734 Consumed Item [5]
60
61 Items consumed: 4
62
63 Consumer 3734 released Semaphore Mutex
64
65 Consumer 3734 released Semaphore Empty
66
67 Consumer 3734 trying to acquire Semaphore Full
68
69 Consumer 3734 successfully acquired Semaphore Full
70
71 Consumer 3734 trying to acquire Semaphore Mutex
72
73 Consumer 3734 successfully acquired Semaphore Mutex
74
75 Consumer 3734 Consumed Item [3]
76
77 Items consumed: 5
78
79 Consumer 3734 released Semaphore Mutex
80

81 Consumer 3734 released Semaphore Empty
82
83 Consumer 3734 trying to acquire Semaphore Full
84
85 Consumer 3734 successfully acquired Semaphore Full
86
87 Consumer 3734 trying to acquire Semaphore Mutex
88
89 Consumer 3734 successfully acquired Semaphore Mutex
90
91 Consumer 3734 Consumed Item [5]
92
93 Items consumed: 6
94
95 Consumer 3734 released Semaphore Mutex
96
97 Consumer 3734 released Semaphore Empty
98
99 Consumer 3734 trying to acquire Semaphore Full
100
101 Consumer 3734 successfully acquired Semaphore Full
102
103 Consumer 3734 trying to acquire Semaphore Mutex
104
105 Consumer 3734 successfully acquired Semaphore Mutex
106
107 Consumer 3734 Consumed Item [6]
108
109 Items consumed: 7
110
111 Consumer 3734 released Semaphore Mutex
112
113 Consumer 3734 released Semaphore Empty
114
115 Consumer 3734 trying to acquire Semaphore Full
116
117 Consumer 3734 successfully acquired Semaphore Full
118
119 Consumer 3734 trying to acquire Semaphore Mutex
120
121 Consumer 3734 successfully acquired Semaphore Mutex
122
123 Consumer 3734 Consumed Item [2]
124
125 Items consumed: 8

126
127 Consumer 3734 released Semaphore Mutex
128
129 Consumer 3734 released Semaphore Empty
130
131 Consumer 3734 trying to acquire Semaphore Full
132
133 Consumer 3734 successfully acquired Semaphore Full
134
135 Consumer 3734 trying to acquire Semaphore Mutex
136
137 Consumer 3734 successfully acquired Semaphore Mutex
138
139 Consumer 3734 Consumed Item [9]
140
141 Items consumed: 9
142
143 Consumer 3734 released Semaphore Mutex
144
145 Consumer 3734 released Semaphore Empty
146
147 Consumer 3734 trying to acquire Semaphore Full
148
149 Consumer 3734 successfully acquired Semaphore Full
150
151 Consumer 3734 trying to acquire Semaphore Mutex
152
153 Consumer 3734 successfully acquired Semaphore Mutex
154
155 Consumer 3734 Consumed Item [1]
156
157 Items consumed: 10
158
159 Consumer 3734 released Semaphore Mutex
160
161 Consumer 3734 released Semaphore Empty
162
163 Consumer Process Ended
