```
/* PROCESS 0 /*

.
.
while (turn != 0)
   /* do nothing */ ;
/* critical section*/;
turn = 1;
.
```

```
/* PROCESS 1 */

.
.
while (turn != 1)
   /* do nothing */;
/* critical section*/;
turn = 0;
.
```

(a) First attempt

```
/* PROCESS 0 */

.
.
while (flag[1])
   /* do nothing */;
flag[0] = true;
/*critical section*/;
flag[0] = false;
.
```

```
/* PROCESS 1 */

.
.
while (flag[0])
   /* do nothing */;
flag[1] = true;
/* critical section*/;
flag[1] = false;
.
```

(b) Second attempt

```
/* PROCESS 0 */

.
.
flag[0] = true;
while (flag[1])
   /* do nothing */;
/* critical section*/;
flag[0] = false;
.
```

```
/* PROCESS 1 */

.
.
flag[1] = true;
while (flag[0])
   /* do nothing */;
/* critical section*/;
flag[1] = false;
.
```

(c) Third attempt

```
/* PROCESS 0 */

.
.
flag[0] = true;
while (flag[1])
{
  flag[0] = false;
  /*delay */;
  flag[0] = true;
}
/*critical section*/;
flag[0] = false;
.
```

```
/* PROCESS 1 */

.
.
flag[1] = true;
while (flag[0])
{
   flag[1] = false;
   /*delay */;
   flag[1] = true;
}
/* critical section*/;
flag[1] = false;
.
```

(d) Fourth attempt

Figure A.1 Mutual Exclusion Attempts

```
/* program barbershop1 */
semaphore max_capacity = 20;
semaphore sofa = 4;
semaphore barber_chair = 3;
semaphore coord = 3;
semaphore cust_ready = 0, finished = 0, leave_b_chair = 0, payment= 0, receipt = 0;

void customer ()
{
   wait(max_capacity);
   enter_shop();
   wait(sofa);
   sit_on_sofa();
   wait(barber_chair);
   get_up_from_sofa();
   signal(sofa);
   sit_in_barber_chair;
   signal(cust_ready);
   wait(finished);
   leave_barber_chair();
   signal(leave_b_chair);
   pay();
   signal(payment);
   wait(receipt);
   exit_shop();
   signal(max_capacity)
}

void barber()
{
   while (true)
   {
      wait(cust_ready);
      wait(coord);
      cut_hair();
      signal(coord);
      signal(finished);
      wait(leave_b_chair);
      signal(barber_chair);
   }
}

void cashier()
{
   while (true)
   {  wait(payment);
      wait(coord);
      accept_pay();
      signal(coord);
      signal(receipt);
   }
}

void main()
{
   parbegin (customer, . . . 50 times, . . . customer, barber, barber, barber, cashier);
}
```

Figure A.5 An Unfair Barbershop

```
/* program barbershop2 */
semaphore max_capacity = 20;
semaphore sofa = 4;
semaphore barber_chair = 3, coord = 3;
semaphore mutex1 = 1, mutex2 = 1;
semaphore cust_ready = 0, leave_b_chair = 0, payment = 0, receipt = 0;
semaphore finished [50] = {0};
int count;

void customer()
{
   int custnr;
   wait(max_capacity);
   enter_shop();
   wait(mutex1);
   custnr = count;
   count++;
   signal(mutex1);
   wait(sofa);
   sit_on_sofa();
   wait(barber_chair);
   get_up_from_sofa();
   signal(sofa);
   sit_in_barber_chair();
   wait(mutex2);
   enqueue1(custnr);
   signal(cust_ready);
   signal(mutex2);
   wait(finished[custnr]);
   leave_barber_chair();
   signal(leave_b_chair);
   pay();
   signal(payment);
   wait(receipt);
   exit_shop();
   signal(max_capacity)
}

void barber()
{
    int b_cust;
    while (true)
    {
        wait(cust_ready);
        wait(mutex2);
        dequeue1(b_cust);
        signal(mutex2);
        wait(coord);
        cut_hair();
        signal(coord);
        signal(finished[b_cust]);
        wait(leave_b_chair);
        signal(barber_chair);
    }
}

void cashier()
{
    while (true)
    {
        wait(payment);
        wait(coord);
        accept_pay();
        signal(coord);
        signal(receipt);
    }
}

void main()
{  count := 0;
   parbegin (customer, . . . 50 times, . . . customer, barber, barber, barber, cashier);
}
```

Figure A.6 A Fair Barbershop