BSBR Producer-Consumer Solution

Producer Process:

```
repeat
  produce item;
  send (dataport, item, localport);
  receive (dataport, ack);
until false;
```

Consumer Process:

```
repeat
  send (reqport, localport);
  receive (localport, item);
  consume item;
until false;
```

BSBR Producer-Consumer Solution

Manager Process:

```
repeat forever
  if count < N then   // can get item from producer
    { receive (dataport, item, port);
      put item in the queue;  count := count + 1;
      send (port, ack);
    }
  if count > 0 then   // can get request from consumer
    { receive (reqport, port);
      take item from queue;  count := count - 1;
      send (port, item);
    }
```

What problem will this cause?

NSNR Producer-Consumer Solution

Producer Process:

```
repeat
  produce item;
  send (dataport, item, localport);
  while not receive (dataport, ack) do nothing;
until false;
```

Consumer Process:

```
repeat
  send (reqport, localport);
  while not receive (localport, item) do nothing;
  consume item;
until false;
```

NSNR Producer-Consumer Solution

Manager Process:

```
repeat forever
  if count = 0 then   // have to get item from producer
    { while not receive (dataport, item, port) do nothing;
      put item in the queue;  count := count + 1;
      send (port, ack);  }
  else if count = N then   // have to send item to consumer
    { while not receive (reqport, port) do nothing;
      take item from queue;  count := count – 1;
      send (port, item);  }
  else   // can either produce or consume
    { if receive (dataport, item, port) then
        { put item in the queue;  count := count + 1;  send (port, ack); }
      if receive (reqport, port) then
        { take item from queue;  count := count – 1; send (port, item); }
    }
```

GC Producer-Consumer Solution

Manager Process:

```
repeat select
  when count < N   // can get item from producer
  { receive (dataport, item);
    put item in the queue;
    count := count + 1;
  }
  when count > 0   // can send item to consumer
  { receive (reqport, port);
    take item from queue;
    send (port, item);
    count := count – 1;
  }
until false;
```

Customer process:

```
send (custport, customer);
take a seat in the entertainment center and wait;
get service from the hairdresser;
send (payport, payment);
```

Hairdresser process:

```
repeat select
   send (dresserport, hairdresser);
   provide hair styling service;
until false;
```

Manager process:

```
numcust = 0;
numserv = 0;

repeat select
   when numcust < M
   { receive (custport, customer);
     numcust = numcust + 1;
     give the customer a seat in the entertainment center;
   }
   when numcust > 0 and numdresser > 0   // gray part: can be omitted
   { receive (dresserport, hairdresser);
     pair customer and dresser;
     numcust = numcust – 1;
     numserv = numserv + 1;
     numdresser = numdresser – 1;
   }
   when numserv > 0
   { receive (payport, payment);
     process payment;
     numserv = numserv – 1;
     numdresser = numdresser + 1;
   }
until salon-close-time;
```