ACK Protocol (... Maybe Semantics...)

SENDER

Sender = wait for input x ∈ M;
Q(x)

Q(x ∈ M) = send x to Receiver;
wait until receipt of msg ACK from Receiver;
Sender

RECEIVER

Receiver = wait until receipt of msg x from Sender;
send ACK to Sender;
output x;
Receiver

ACK/NACK Protocol

SENDER

```
Sender      =   wait for input x ∈ M;
                Q(x)

Q(x ∈ M)    =   send x to Receiver;
                wait until receipt of msg y from Receiver;
                if y == ACK then Sender
                else if y == NACK then Q(x)
```

Receiver

```
Receiver    =   wait until receipt of msg x from Sender;
                if x ∈ M then {
                        send ACK to Sender;
                        output x;
                        Receiver}
                else if x ∈ M' then {
                        send NACK to Sender;
                        Receiver}
```

Polling protocol

RECEIVER

```
Receiver    =    send msg POLL to Sender;
                 Wait

Wait        =    wait until receipt of msg x from Sender;
                 if x ∈ M then {
                         output x;
                         Receiver}
                 else if x ∈ M' then {
                         send REPT to Sender;
                         Wait}
```

SENDER

```
Sender      =    Q(a)                                  // a is a constant

Q(x ∈ M)    =    wait until receipt of msg y from Receiver;
                 if y == POLL then {
                         wait for input x ∈ M from user;
                         send x to Receiver;
                         Q(x)}
                 else if y == REPT then {
                         send x to Receiver;
                         Q(x)}
```

ACK/NACK + TIMEOUT (... At-Least-Once Semantics...)

SENDER

```
Sender      =   wait for input x ∈ M;
                Q(x)

Q(x ∈ M)    =   send x to Receiver;
                start timer(s);                          // set timeout after s seconds
                wait until (receipt of msg y from Receiver OR timeout);
                if receipt of msg y from Receiver then {
                        if y == ACK then {
                                reset timer;
                                Sender}
                        else if y == NACK then {
                                reset timer;
                                Q(x)}}
                else {                                   // timeout
                        reset timer;
                        Q(x)}
```

RECEIVER

```
Receiver    =   wait until receipt of msg x from Sender;
                if x ∈ M then {
                        send ACK to Sender;
                        output x;
                        Receiver}
                else if x ∈ M' then {
                        send NACK to Sender;
                        Receiver}
```

PAR Protocol (ACK + TIMEOUT + NUMBERING SCHEME)

SENDER

```
Sender        =    S(1)

S(n ∈ N0)     =    wait for input x ∈ M;
                   Q(n, x)

Q(n ∈ N0, x ∈ M)  =   send (n, x) to Receiver;
                      start timer(s);
                      wait until (receipt of msg y from Receiver OR timeout);
                      if receipt of msg y == ACK from Receiver then {
                            reset timer;
                            S(succ(n))}
                      else {                                  // timeout
                            reset timer;
                            Q(n, x)}
```

RECEIVER

```
Receiver      =    R(0)

R(n ∈ N0)     =    wait until receipt of msg (i, x) from Sender;
                   if x ∈ M then {
                         if i = succ(n) then {                // expected msg
                               send ACK to Sender;
                               output x;
                               R(succ(n))}
                         elseif i = n then {                  // same msg
                               send ACK to Sender;
                               R(n)}
                         else R(n)}
                   else if x ∈ M' then R(n)
```

ACK/NACK + TIMEOUT + NUMBERING SCHEME (... At-Most-Once Semantics...)

... Exercise...

PAR Protocol + TIMEOUT + NUMBERING SCHEME + NUMBERED ACK

SENDER

```
Sender      =   S(1)

S(n ∈ N0)   =   wait for input x ∈ M;
                Q(n, x)

Q(n ∈ N0, x ∈ M)  =  send (n, x) to Receiver;
                     start timer(s);
                     wait until receipt of msg a from Receiver OR timeout;
                     if receipt of msg a ∈ N0 from Receiver then {
                             if (a == n) then {                  // expected ACK
                                     reset timer;
                                     S(succ(n))}
                             else {                              // different ACK
                                     reset timer;
                                     Q(n, x)}}
                     else {                                      // timeout or a ∉ N0
                             reset timer;
                             Q(n, x)}
```

RECEIVER

```
Receiver    =   R(0)

R(n ∈ N0)   =   wait until receipt of msg (i, x) from Sender;
                if x ∈ M then {
                        if i = succ(n) then {                    // expected msg
                                send i to Sender;
                                output x;
                                R(i)}
                        else {
                                send n to Sender;
                                R(n)}}
                else if x ∈ M' then R(n)
```

Polling protocol + TIMEOUT + NUMBERING SCHEME

... Exercise...

Two-Way Exchange (or Handshake) Protocol

Req: requests;
Accept: positive replies;
Refuse: negative replies;
Accept and Refuse are DISJOINT
ref, ERROR ∈ Refuse (internal message indicating refusal).

At **(. . .)**, both parties are sufficiently finished to go on with the next part of their tasks.

SENDER

```
Sender      =   wait for input x ∈ Req;
                send x to Receiver;
                start timer(s);
                SR

SR          =   wait until receipt of msg y from Receiver OR timeout;
                if receipt of msg y from Receiver then {
                        if (y ∈ Accept) then {          // y ∈ Accept
                                reset timer;
                                output y;
                                (...)}
                        else {                          // y ∈ Refuse
                                reset timer;
                                output y;
                                Sender}}
                else {                                  // timeout
                        reset timer;
                        output ERROR;                   // ERROR ∈ Refuse
                        Sender}
```

RECEIVER

```
Receiver    =    wait until receipt of msg y ∈ Req from Sender;
                 output y;
                 start timer(s);
                 RR

RR          =    wait for input x OR timeout;
                 if receipt of input x then {
                        if (x ∈ Accept) then {          // x ∈ Accept
                               reset timer;
                               send x to Sender;
                               (...)}
                        else {                          // x ∈ Refuse
                               reset timer;
                               send x to Sender;
                               Receiver}}
                 else {                                 // timeout
                        reset timer;
                        output ERROR;                   // ERROR ∈ Refuse
                        send ERROR to Sender;
                        Receiver}
```

Three-Way Handshake Protocol

SENDER

```
Sender      =   wait for input r ∈ Req;
                send (x, r) to Receiver;                    // x ∈ TypeOK
                start timer(s);
                SR(x)

SR(x ∈ TypeOK)      =   wait until receipt of msg y from Receiver OR timeout;
                        if receipt of msg y = (p, q, c) from Receiver then {
                            if ((c ∈ Accept) & (p, q ∈ TypeOK)) then {
                                reset timer;
                                if (p = x) then {
                                    send (p, q, check) to Receiver;
                                    output c;
                                    (...)}
                                else {
                                    output ERROR;
                                    Sender}}
                            elseif ((c ∈ Refuse) & (p, q ∈ TypeOK)) then {
                                if (p = x) then {
                                    reset timer;
                                    output c;
                                    Sender}
                                else
                                    SR(x)}}
                        else {                                  // timeout
                            reset timer;
                            output ERROR;                   // ERROR ∈ Refuse
                            Sender}
```

RECEIVER

```
Receiver      =     wait until receipt of msg y = (x ∈ TypeOK, r ∈ Req) from Sender;
                    output r;
                    start timer(s);
                    RR(x)

RR (x ∈ TypeOK)   =     wait for input c or timeout;
                        if receipt of input c then {
                                if (c ∈ Accept) then {
                                        send (x, y, c) to Sender;     // y ∈ TypeOK
                                        RC(x, y)}
                                elseif (c ∈ Refuse) then {
                                        send (x, y, c) to Sender;
                                        Receiver}}
                        else {                                        // timeout
                                reset timer;
                                send (x, y, ERROR) to Sender          // ERROR ∈ Refuse
                                output ERROR;
                                Receiver}

RC (x, y ∈ TypeOK) =    wait until receipt of msg y = (p ∈ TypeOK, q ∈ TypeOK, c ∈
                        Check) from Sender OR timeout;
                        if receipt of msg y = (p, q, c) then {
                                if (p = x) & (q = y) then {
                                        reset timer;
                                        (...)}
                                else RC(x, y)}
                        else {                                        // timeout
                                reset timer;
                                output ERROR;
                                Receiver}
```