1. 

A. $S = 2w$, Receiver thinks $i = i-2w$?
   If the sender is sending packet $(i)$. Because $S = 2w$, the lower boundary of sender's window must be $(i-w+1)$, Receiver think packet$(i)$ as packet$(i-2w)$, but packet$(i-2w)$ is not inside the possible sender's window, therefore it couldn't happen.

   ![Diagram of window boundaries]

B. $S = 2w$, Receiver thinks $i = i+2w$?
   If the sender is sending packet $(i)$. Because $S = 2w$, the upper boundary of sender's window must be $(i+w-1)$, Receiver think packet$(i)$ as packet$(i+2w)$, but packet$(i+2w)$ is not inside the possible sender's window, therefore it couldn't happen.

   ![Diagram of window boundaries]

C. $S = w+1$, Receiver thinks $i=i+w+1$?
   If the sender is sending packet $(i)$. Because $S = w+1$, the upper boundary of sender's window must be $(i+w-1)$, Receiver think packet$(i)$ as packet$(i+w+1)$, but packet$(i+w+1)$ is not inside the possible sender's window, therefore it couldn't happen.

   ![Diagram of window boundaries]

D. $S = w+1$, Receiver thinks $i = i-w-1$?
   If the sender is sending packet $(i)$. Because $S = w+1$, the lower boundary of
sender's window must be \((i-w+1)\), Receiver think packet\((i)\) as packet\((i-w-1)\), but packet \(i-w-1\) is not inside the possible sender's window, therefore it couldn't happen.

\[
\begin{array}{ccc}
i-w-1 & i-w+1 & i \\
\end{array}
\]

E. No matter \(S\) is increasing from \(2w\) and \(w\), the range of sender's window is still lies on \([i-w+1, i+w-1]\), so that \(S \geq 2w, S > w\) will hold for all cases.

2.
(a) SR, \(S = 5\), \(W = 3\)

\[
\begin{array}{cccccccc}
0 & 1 & 2 & 3 & 4 & 0 & 1 & 2 \\
0 & 1 & 2 & 3 & 4 & 0 & 1 & 2 \\
0 & 1 & 2 & 3 & 4 & 0 & 1 & 2 \\
0 & 1 & 2 & 3 & 4 & 0 & 1 & 2 \\
0 & 1 & 2 & 3 & 4 & 0 & 1 & 2 \\
0 & 1 & 2 & 3 & 4 & 0 & 1 & 2 \\
\end{array}
\]

// Timeout resend
\[
\begin{array}{cccccccc}
0 & 1 & 2 & 3 & 4 & 0 & 1 & 2 \\
0 & 1 & 2 & 3 & 4 & 0 & 1 & 2 \\
0 & 1 & 2 & 3 & 4 & 0 & 1 & 2 \\
0 & 1 & 2 & 3 & 4 & 0 & 1 & 2 \\
0 & 1 & 2 & 3 & 4 & 0 & 1 & 2 \\
0 & 1 & 2 & 3 & 4 & 0 & 1 & 2 \\
\end{array}
\]

// Mix up, receiver think this is the new pkt(0)
(b) Go-Back-N, $S = 5$, $W = 5$

// Timeout resend

// Mix up, receiver think this is the new pkt(2)
3. FSM (T: timeout), there is more than one way to do this problem.

\[ r_2(0) | r_1(1) | r_2(1) \quad r_2(1) | r_1(0) | r_2(0) \]

\[ \frac{t}{\sigma(0)} \quad \frac{t}{\sigma(1)} \]

\[ \text{Wait u1Ack}(0) \quad \text{Wait u1Ack}(1) \]

\[ r_2(0) \quad r_1(0) \quad r_2(0) \quad r_1(0) \]

\[ \frac{r_2(0)}{S(1)} \quad \frac{r_1(0)}{S(0)} \quad \frac{r_2(1)}{S(1)} \quad \frac{r_1(1)}{S(0)} \]

\[ \text{Wait u1 or u2Ack}(0) \quad \text{Wait u1 or u2Ack}(1) \]

\[ r_2(1) | r_1(1) \quad r_2(1) | r_1(1) \]

\[ \frac{r_2(1)}{S(1)} \quad \frac{r_1(1)}{S(0)} \quad \frac{r_2(1)}{S(1)} \quad \frac{r_1(1)}{S(0)} \]

\[ \frac{t}{\sigma(0)} \quad \frac{t}{\sigma(1)} \]

\[ \text{Wait u2Ack}(0) \quad \text{Wait u2Ack}(1) \]

\[ r_1(0) \quad r_2(1) | r_1(1) \quad r_1(0) | r_2(0) \]
A codeword of degree L has L data words XOR'd together, and can recover a codeword only if (L-1) of the data forming the codeword are already received (in the buffer) and one has not.

If \( L > k+1 \), then the probability is 0.

Otherwise,
\[
P = \frac{(N-k) \cdot \binom{k}{L-1}}{\binom{N}{L}}
\]
\[
= \frac{(N-k) \cdot (N-L)! \cdot L! \cdot k!}{(k-L+1)! \cdot (L-1)! \cdot N!}
\]
\[
= \frac{(N-k) \cdot (N-L)! \cdot L! \cdot k!}{(k-L+1)! \cdot N!}
\]

where \((x \text{ choose } y) = x! / y!(x-y)!\), \(L\) means the "l" letter in the document.

5.

A. \(N\) is a random variable that equals the number of packets that need to be sent in order to receive \(k\) data packet.
And \(E[N]\) is the expected numbers of data plus repair packets that are sent to recover the \(k\) data packets.
\[
E[N] = \sum_{i=k}^{\infty} i \cdot p(N = i)
\]
\[
= \sum_{i=k}^{\infty} i \cdot \binom{i-1}{k-1} \cdot (1-p)^k \cdot p^{i-k}
\]

Where \(\binom{i-1}{k-1}\) = Choose \(k-1\) from \(i-1\) = \(\frac{(i-1)!}{(k-1)! \cdot (i-k)!}\)

B.
If \(n < k\), then
\[
p(N \leq n) = 0
\]
If \(n \geq k\), then
\[
p(N \leq n) = \sum_{i=k}^{n} \binom{i-1}{k-1} \cdot (1-p)^k \cdot p^{i-k}
\]
for each \(i\), the assumption is that the last packet is received in the \(i\)th round. So you have that last \((k\) th) arrived packet in the \(i\)th slot, and you have to "choose" where to put the remaining \(k-1\) packets in the other \(i-1\) slots.