1. 使用 UDP 中校验和的计算方法计算下面三个 16 位二进制数值的校验和（给出计算过程）

   1011010011101000

   0110111011000111

   1110011100111000

   答案：

   (1) **首先计算前两个二进制数的和**

   前两个二进制数的和为 17 位二进制数 10010001110101111，有溢出，进行回卷操作（将进位加至结果的最低位），得到结果 0010001110110000。

   (2) **将上一步得到的结果与题中第三个二进制数相加**

   上一步的结果 0010001110110000，第三个二进制数是 1110011100111000，它们相加后的结果为 17 位二进制数 10000101011101000，有溢出，进行回卷操作（将进位加至结果的最低位），得到结果 0000101011101001。

   (3) **对上一步的求和结果取反，得到最终的头部校验和值**

   上一步的结果 0010001110110000。

   最终结果：1101110001001111。

2. 在城市 A 和城市 B 之间有一条 Internet 主干网链路，其数据率为 1 Gb/s，往返时间（RTT）为 100 ms，城市 A 中的一台主机通过 TCP 连接向城市 B 中的一台主机发送数据，接收端通告的窗口从未大于 1 MB，那么发送端可以达到的最大吞吐率是多少?

   答案：

$$\text{数据传输时间} = \frac{\text{接收端通告的窗口}}{\text{数据率}} = \frac{1MB}{1Gbps} = 0.008s$$

$$\text{最大吞吐率 (Throughput)} = \frac{\text{接收端通告的窗口}}{\text{数据传输时间 + RTT}} = \frac{1MB}{0.008s + 0.1s} = 74.074Mbps$$

3. 分析下面捕获的 TCP 报文片段，请回答如下问题：

TCP 1026>http[ACK]Seq=51231 Ack=1 Win=65535 Len=1460

TCP 1026>http[ACK]Seq=52691 Ack=1 Win=65535 Len=1460

TCP 1026>http[ACK]Seq=54151 Ack=1 Win=65535 Len=1460

TCP http>1026[ACK]Seq=1 Ack=51231 Win=62780 Len=0

TCP 1026>http[ACK]Seq=55611 Ack=1 Win=65535 Len=1460

TCP 1026>http[PSH,ACK] Seq=57071 Ack=1 Win=65535 Len=892

TCP http>1026[ACK] Seq=1 Ack=52691 Win=62780 Len=0

TCP [TCP Dup ACK 98#1] http>1026 [Ack]Seq=1 Ack=52691 Win=62780

TCP [TCP Dup ACK 98#2] http>1026 [Ack]Seq=1 Ack=52691 Win=62780

TCP TCP 1026>http[ACK] Seq=52691 Ack=1 Win=65535 Len=1460

TCP TCP 1026>http[ACK] Seq=55611 Ack=1 Win=65535 Len=1460

(a) 请问哪些是重传报文（写出其发送序列号），重传的原因分别是什么?

答案:

TCP TCP 1026>http[ACK] Seq=52691 Ack=1 Win=65535 Len=1460

TCP TCP 1026>http[ACK] Seq=55611 Ack=1 Win=65535 Len=1460

都是超时重传。

(b) ACK 报文中 Win 字段的作用是什么?

答案:

ACK 报文中 Win 字段用于流量控制，用于指示接收方愿意接收的字节数量。

(c) 当接口层为不可靠的无线链路时（出错率较高），TCP 的拥塞控制机制对网络性能有何影响? 简单进行解释。

答案:

当接口层为不可靠的无线链路时（出错率较高），可能会经常出现报文段丢失的情况。

一个丢失的报文段意味着拥塞。丢失报文段时，TCP 发送方会减小它的拥塞窗口长度，即降低其发送速率。拥塞窗口可能一直保持在一个较低的水准，TCP 发送方的发送速率会小于带宽，导致不能充分利用带宽，这条 TCP 连接的平均吞吐量较低。

4. 两台主机 A 和 B，主机 A 上运行的 Web 服务器进程试图向主机 B 上的浏览器进程发送数据。对于每个 TCP 连接，主机 A 上的 TCP 维护一个 512 字节的发送缓存，主机 B 上的 TCP 维护一个 1024 字节的接收缓存。为了简单起见，假设 TCP 序列号从 0 开始。

(a) 主机 B 的 TCP 层从主机 A 按顺序接收到第 560 字节，浏览器进程只从中读出前 60 字节，那么在主机 B 发送给主机 A 的 TCP 段首部中的确认序列号（ACK#）和接收窗口大小（RcvrWindow Size）分别为多少?

答案:

(1) 主机 B 填充进报文段的确认号是主机 B 期望从主机 A 收到的下一字节的序号。主机 B 的 TCP 层从主机 A 按顺序接收到第 560 字节，故主机 B 发送给主机 A 的 TCP 段首部中的确认序列号为 561。

(2) 定义以下变量:

- RcvBuffer: 接收缓存的大小。
- rwnd: 接收窗口。
- LastByteRead: 主机 B 上的应用进程从缓存中独处的数据流的最后一个字节的编号。
- LastByteRcvd: 从网络中到达的并且已放入主机 B 接收缓存中的数据流的最后一个字节的编号。

rwnd 的设置公式如下:

$$rwnd = RcvBuffer - [LastByteRcvd - LastByteRead]$$

由题意得，RcvBuffer=1024，LastByteRcvd=560，LastByteRead=60。解得 rwnd=524。故主机 B 发送给主机 A 的 TCP 段首部中的接收窗口大小为 524。

最终结果如下:

主机 B 发送给主机 A 的 TCP 段首部中的确认序列号为 561。

主机 B 发送给主机 A 的 TCP 段首部中的接收窗口大小为 524。

(b) 在同一个 TCP 连接中，如果主机 A 的拥塞窗口设置成 1 个 MSS（Maximum Segment Size，536 字节），主机 B 通告的流控窗口为 560 字节，主机 A 从主机 B 接收到的最后确认序列号为第 700 字节，主机 A 发送给主机 B 的最后字节为 900。

i. 假设主机 A 没有再收到 ACK，它的窗口大小没有改变，那么主机 A 能够发送的最大字节号是多少?

答案：定义以下变量：

- cwnd: 拥塞窗口大小。
- rwnd: 接收窗口。
- LastByteSent: 主机 A 发送给主机 B 的最后字节。
- LastByteAcked: 主机 A 从主机 B 接收到的最后确认序列号。

在一个发送方中未被确认的数据量不会超过拥塞窗口 cwnd 的值，也不会超过接收方通告的流控窗口值，即

$$LastByteSent - LastByteAcked \leq min(cwnd_A, rwnd_B) = 536$$

由题意，$LastByteAcked = 700, cwnd = 536 \leq$ 主机 B 通告的流控窗口 $= 560$。代入得到 $LastByteSent \leq 1236$。**忽略以上内容**。

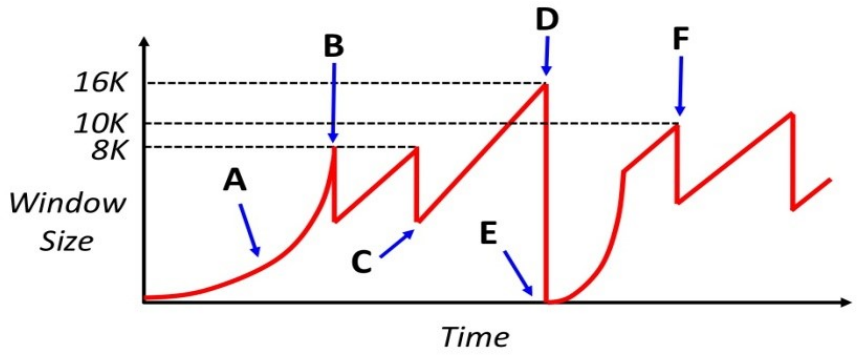
故主机 A 能够发送的最大字节号是 900+311=1211。原因和下一小题一样

ii. 假设主机 A 没有再收到另外的 ACK，则运行在主机 A 上的 Web 服务器进程在阻塞前可以再向 Socket 写入多少字节?

答案：

由已知得，主机 A 从主机 B 接收到的最后确认序列号为第 700 字节，主机 A 发送给主机 B 的最后字节为 900。所以主机 A 的发送缓冲区已被占用 $900 - 700 + 1 = 201$ 字节，主机 A 发送缓存区大小为 512 字节，所以**主机 A 上的 Web 服务器进程在阻塞前可以再向 Socket 写入** $512 - 201 = 311$ **字节**。311 字节小于主机 A 的拥塞窗口（536 字节），可以正常发送。

5. 如下图所示，纵轴表示 TCP 拥塞窗口大小，横轴为时间轴。请根据下图回答如下问题：



(a) 在图中 B、D 处分别发生了什么事件，B、D 事件的发生表明在网络中一定有数据包被丢弃吗?

答案：

B 处发生的事件：出现了一个由三个冗余的 ACK 指示的丢包事件。

D 处发生的事件：出现了一个由超时指示的丢包事件（即拥塞）。TCP 发送方将 TCP 拥塞窗口 *cwnd* 设置为 1 MSS 并进入慢启动状态。将第二个状态变量的值 ssthresh（“慢启动阈值”的速记）设置为 *cwnd*/2，即当检测到拥塞时将 ssthresh 置为拥塞窗口值的一半。

B、D 事件的发生不能表明在网络中一定有数据包被丢弃。比如，考虑一种情况，接收方虽然收到了发送方发来的数据包，但是由于某些原因一直没有发送 ACK 给发送端，TCP 发送方的等待时间已经超过了 TCP 发送方计时器指定的时间。在这种情况下，并没有数据包在网络中被丢弃。

(b) 考虑图中 A 段曲线，为什么 TCP 拥塞窗口采取此种增长方式而非线性增长?

答案：

当一条 TCP 连接开始时（或者 TCP 拥塞窗口 cwnd 设置为 1 时），TCP 进入慢启动状态。慢启动算法采用指数增长方式增长 TCP 拥塞窗口。

如果 TCP 发送方过于谨慎，对于 TCP 拥塞窗口采用采取线性增长而非指数增长，**会导致发送太慢，不能充分利用网络的带宽**。

(c) 假设发送端在 $t = 0$ 时刻开始建立了一个 TCP 连接，TCP 连接的 MSS 为 1000 字节，发送端到接收端的往返延时（RTT）为 100 ms。那么到达 B、C、D、F 点所用的时间分别为多少？（假设发送端有充足的数据等待发送）

答案：

由图中看出各点对应的 TCP 拥塞窗口 cwnd 分别为 $cwnd_B = 8KB, cwnd_C = 4KB, cwnd_D = 16KB, cwnd_F = 10KB$

在慢启动状态，每收到一个新的 ACK，拥塞窗口就增加 1 个 MSS。故此时有：

$$cwnd_B = 2^{\frac{t_B - 0}{RTT}} * MSS$$

$$t_B = RTT * \log_2 \frac{cwnd_B}{MSS}$$

从点 $t = 0$ 到点 B，“慢启动状态”，发送端不断收到新的 ACK，**故可解出 B 点时间为** $0.3s$。

B 点时，发送端收到了 3 个冗余的 ACK，慢启动阈值标记 $ssthresh = \frac{cwnd}{2} = 4KB$，拥塞窗口更新 $cwmd = ssthresh + 3 * MSS = 7KB$，进入“快速恢复”状态。进入这个状态后存在下列 3 种情况：

1. 再次收到重复的 ACK，每收到一个重复 ACK，cwnd 更新为 $cwnd = cwnd + MSS$，之后仍处于“快速恢复”状态，cwnd 指数增长；
2. ACK 超时，进入慢启动状态，ssthresh 和 cwnd 都更新，$ssthresh = \frac{cwnd}{2}$ $cwnd = 1$；
3. 收到了新的 ACK，cwnd 更新为 $cwnd = ssthresh$。

观察 B 点到 C 点的线段，首先排除第 1 种情况，因为这段时间里的 cwnd 是线性增长的。再排除第 2 种情况，因为如果在这段时间里出现 ACK 超时，cwnd 会降至 1 个 MSS。所以，在进入“快速恢复”状态后，在足够短的时间内，发送端收到了新的 ACK，进入了“拥塞避免”状态，cwnd 更新为 $cwnd = ssthresh = 4KB$。进入“拥塞避免”状态，每收到一个新的 ACK，$cwnd$ 的值增加一个 MSS。故此时有：

$$cwnd_C * 2 - \frac{cwnd_B}{2} = \frac{t_C - t_B}{RTT} * MSS$$

$$t_C = \frac{cwnd_C * 2 - \frac{cwnd_B}{2}}{MSS} * RTT + t_B$$

从点 B 到点 C 处于“拥塞避免”状态，发送端不断收到新的 ACK，**故可解出 C 点时间为** $0.7s$。

C 点出现的事件与 B 点原理相同。

同理，从点 C 到点 D 处于“拥塞避免”状态，**故可解出 D 点时间为** $1.9s$。

D 点后先进入了“慢启动”状态，在达到慢启动阈值 $ssthresh = \frac{cwnd_D}{2} = 8KB$ 后，进入“拥塞避免”状态。同理，**可解出 F 点时间为** $2.4s$。

总结： $time_B = 0.3s, time_C = 0.7s, time_D = 1.9s, time_F = 2.4s$。