1、

(a) How can a new *SampleRTT* be obtained in TCP?

报文段的样本 RTT（表示为 SampleRTT）就是从某报文段被发出（即交给 IP）到对该报文段的确认被收到之间的时间量。大多数 TCP 的实现仅在某个时刻做一次 SampleRTT 测量，而不是为每个发送的报文段测量一个 SampleRTT。也就是说，在任意时刻，仅为一个已发送的但目前尚未被确认的报文段估计 SampleRTT，从而产生一个接近每个 RTT 的新 SampleRTT 值。另外，TCP 决不为已重传的报文段计算 SampleRTT，而仅为传输一次的报文段测量 SampleRTT。

(b) Suppose that the initial *EstimatedRTT* is 10, and that TCP gets the following sequence of *SampleRTT* (in ms)

10, 20, 10, 10, 10, 10, 10, 20, 20, 20, 20

- **If α=0.9, compute the sequence of *EstimatedRTT* obtained from the above sequence of *SampleRTT.***

  EstimatedRTT 1=0.9*10+0.1*10=10
  EstimatedRTT 2=0.9*10+0.1*20=11
  EstimatedRTT 3=0.9*11+0.1*10=10.9
  EstimatedRTT 4=0.9*10.9+0.1*10=10.81
  EstimatedRTT 5=0.9*10.81+0.1*10=10.73
  EstimatedRTT 6=0.9*10.73+0.1*10=10.66
  EstimatedRTT 7=0.9*10.66+0.1*10=10.59
  EstimatedRTT 8=0.9*10.59+0.1*20=11.53
  EstimatedRTT 9=0.9*11.53+0.1*20=12.38
  EstimatedRTT 10=0.9*12.38+0.1*20=13.14
  EstimatedRTT 11=0.9*13.14+0.1*20=13.83

- **Repeat the question if α=0.5, and comment on the pros and cons of choosing α=0.5 versus α=0.9.**

  EstimatedRTT 1=0.5*10+0.5*10=10
  EstimatedRTT 2=0.5*10+0.5*20=15

EstimatedRTT 3=0.5*15+0.5*10=12.5
EstimatedRTT 4=0.5*12.5+0.5*10=11.25
EstimatedRTT 5=0.5*11.25+0.5*10=10.63
EstimatedRTT 6=0.5*10.63+0.5*10=10.32
EstimatedRTT 7=0.5*10.32+0.5*10=10.16
EstimatedRTT 8=0.5*10.16+0.5*20=15.08
EstimatedRTT 9=0.5*15.08+0.5*20=17.54
EstimatedRTT 10=0.5*17.54+0.5*20=18.77
EstimatedRTT 11=0.5*18.77+0.5*20=19.39

EstimatedRTT 是一个由 SampleRTT 值而得出的加权平均值，根据每次得到的新的 RTT 样本不同呈现不同的变化趋势。加权平均对最新样本赋予的权值要大于对老样本赋予的权值。公式中通过设定α的大小，可以改变最新样本的权重值大小。

好处：对于$\alpha = 0.5$ 这种情况，最新样本的权重增大，对 RTT 突然稳定上升或者突然稳定下降有较好的预测能力。如本例中 SampleRTT 序列呈明显稳定上升趋势，SampleRTT[10] = 20,而$\alpha = 0.5$ 时，用前九个 SampleRTT 预测得到的 EstimatedRTT[10] =17.5390625，较$\alpha = 0.9$ 时估计的 12.3782969 要接近很多。

弊端：$\alpha = 0.5$ 较α=0.9 而言，对于 SampleRTT 无规律波动过大的情况，不能较有效地反应累计的平均变化情况。

(c) Suppose now the sequence of *SampleRTT* (in ms) is 10, 6, 14, 6, 14, 10, 10, 25, 20, 20, 16, 20. What is the problem of applying the above procedure for computing the TCP retransmission timeout? How does TCP solve the problem?

由于β是个常量，即 RTO＝β *RTT (β通常大于 1，此处取$\beta=2$)。

RTO 应该大于等于 EstimatedRTT，否则将会造成不必要的重传。但是 RTO 也不能比 EstimatedRTT 大太多，否则当报文丢失时，TCP 不能很快地重传该报文段，从而给上层应用带来很大的数据传输时延。因此，要求将超时时间间隔设为 EstimatedRTT 加上一定的余量。如果选取 β 值过大，则会造成丢包后等待时间过长；如果 β 值过小，则可能使 RTO<RTT，造成提前超时。

以下 RTT 计算方法同(a)题，即$\alpha=0.9$，初始 EsitimatedRTT=0。下表中，RTO＜EstmatedRTT，会造成重传的情况，用红色标记。RTO 比 EstimatedRTT 大太多，当报文丢失不能很快重传的情况，用黄色标记。

i	EstimatedRTT[i]	RTO	SampleRTT
2	10	20	6
3	9.6	19.2	14
4	10.04	20.08	6
5	9.636	19.272	14
6	10.0724	20.1448	10
7	10.06516	20.13032	10
8	10.058644	20.117288	25
9	11.5527796	23.1055592	20

10	12.39750164	24.79500328	20
11	13.15775148	26.31550296	16
12	13.44197633	26.88395266	20
13	14.0977787	28.1955574	

TCP 在处理上述问题的过程中，引入了 RTT 偏差 Deviation，用于估算 SampleRTT 一般会偏离 EstimatedRTT 的程度：

Difference = SampleRTT - EstimatedRTT

EstimatedRTT = EstimatedRTT + (δ *Difference)

= (1- δ) *EstimatedRTT+ δ * SampleRTT

Deviation= Deviation+ δ * (|Difference| - Deviation)

= (1- δ) *Deviation+ δ * |Difference|

RTO = μ * EstimatedRTT + ϕ * Deviation

(μ ≈1, ϕ≈4)

如果 SampleRTT 值波动较小，那么 Deviation 的值就会很小；相反，如果波动很大，那么 Deviation 的值就会很大。

下面计算中，设初始 EstimatedRTT=10，初始 Deviation=0.

i	EstimatedRTT[i]	Deviation	RTO(原)	RTO（新）	Sample RTT
2	10	0	20	10	6
3	9.6	0.4	19.2	11.2	14
4	10.04	0.8	20.08	13.24	6
5	9.636	1.124	19.272	14.132	14
6	10.0724	1.448	20.1448	15.8644	10
7	10.06516	1.31044	20.13032	15.30692	10
8	10.058644	1.185912	20.117288	14.802292	25

9	11.5527796	2.5614564	23.1055592	21.7986052	20
10	12.39750164	3.1500328	24.79500328	24.99763284	20
11	13.15775148	3.595279356	26.31550296	27.5388689	16
12	13.44197633	3.519976273	26.88395266	27.52188142	20
13	14.0977787	3.823781013	28.1955574	29.39290275	

改进的算法能够综合反映累计偏差，所以更能够恰当地估计RTO，有效地减少提前超时和丢包等待过长。

2、

（写出所给数据的算术校验和计算过程即可）

AEF5 ---- 1010 1110 1111 0101
D698 ---- 1101 0110 1001 1000
FE33 ---- 1111 1110 0011 0011

```
  1010 1110 1111 0101
  1101 0110 1001 1000
-------------------------
1 1000 0101 1000 1101
回卷                    1
-------------------------
  1000 0101 1000 1110
  1111 1110 0011 0011
-------------------------
1 1000 0011 1100 0001
回卷                    1
-------------------------
  1000 0011 1100 0010
取反
-------------------------
  0111 1100 0011 1101
```

checksum= 0111 1100 0011 1101=7C3D