Exam 1

1. Bounded Timestamping System

If concurrently executed, the state of three threads could be in (00,11,22) where there is no precedence. Or, (10,11,12) where all the threads are in one ring, again no precedence.

2. LL/SC 의 consensus number

We can construct CAS using LL/SC. Therefore it's infinity

3. SRSW, MV, Atomic register

You need 128 registers.

4. multiple consensus using CAS

It could be possible to reuse the register if the new base value can be set differently from all the values that can be written in the previous consensus. Otherwise, ABA problem is always possible.

5. multiple consensus using LL/SC
Using LL/SC, ABA problem would disappear.
However, the forward progress might not be guaranteed.

6. Filter's algorithm

While thread 0 is waiting in some level, it is possible for two other threads to enter the same level and pass the thread 0 repeatedly if thread 0 is slow.

7. 4-thread consensus using atomic 2-assignment

It's impossible. What happens if both the winners of the the first phase dies right after they win?
Nobody will know what will happen.

Exam 2

1. Lock-free 와 Wait-free 의 비교

Lock free means that some thread is making progress no matter what.
Wait-free means that all the threads should make progress arbeit slow.

2. Hopscotch Hashing

1	2	3	4	5	6	7	8	9	10
A	E	D		C		B			
1000	1000	1000		1000		1000			

1) F

1	2	3	4	5	6	7	8	9	10
A	E	F	D	C		B			
1000	1000	1100	0000	1000		1000			

2) G

1	2	3	4	5	6	7	8	9	10
A	E	F	D	C	G	B			
1000	1000	1100	0000	1000	1000	1000			

1	2	3	4	5	6	7	8	9	10
A	E	F	D		G	B	C		
1000	1000	1100	0000	0001	1000	1000	0000		

1	2	3	4	5	6	7	8	9	10
A	E	F		D	G	B	C		
1000	1000	1010	0000	0001	1000	1000	0000		

1	2	3	4	5	6	7	8	9	10
A	E	F	H	D	G	B	C		
1000	1000	1010	1000	0001	1000	1000	0000		

1	2	3	4	5	6	7	8	9	10
A	E	F	H	D	G		C	B	
1000	1000	1010	1000	0001	1000	0010	0000	0000	

1	2	3	4	5	6	7	8	9	10
A	E	F	H	D	G	I	C	B	
1000	1000	1010	1000	0001	1000	1010	0000	0000	

5) J

1	2	3	4	5	6	7	8	9	10
A	E	F	H	D	G		C	B	I

1000	1000	1010	1000	0001	1000	0011	0000	0000	0000

1	2	3	4	5	6	7	8	9	10
A	E	F		D	G	H	C	B	I
1000	1000	1010	0001	0001	1000	0011	0000	0000	0000

1	2	3	4	5	6	7	8	9	10
A		F	E	D	G	H	C	B	I
1000	0010	1010	0001	0001	1000	0011	0000	0000	0000

1	2	3	4	5	6	7	8	9	10
A	J	F	E	D	G	H	C	B	I
1000	1010	1010	0101	0001	1000	0011	0000	0000	0000

3. RCU 와 Read/Write Lock 비교

RCU allows the readers to proceed even when an update is being made.
In RW lock, only one writer or multiple readers can proceed at a time.

4. Quick Sort

The worst case

$$B_1(n) = B_1(n-1) + A_1(n) = B_1(n-1) + \Theta(n)$$
$$B_1(n) = \Theta(n^2)$$

.

$$B_{inf}(n) = B_{inf}(n-1) + A_{inf}(n) = B_{inf}(n-1) + \Theta(\log n)$$
$$B_{inf}(n) = \Theta(n \log n)$$

The best case

$$C_1(n) = C_1\left(\frac{n}{2}\right) + A_1(n) = C_1\left(\frac{n}{2}\right) + \Theta(n)$$

$$C_1(n) = \Theta(n \log n)$$

$$C_{inf}(n) = C_{inf}\left(\frac{n}{2}\right) + A_{inf}(n) = C_{inf}\left(\frac{n}{2}\right) + \Theta(\log n)$$
$$C_{inf}(n) = \Theta(\log^2 n).$$

5. Combining Tree

: A:0, B:12, C:2, D:13, E:9, F:5, G:14, H:11.

Thread	A	B	C	D	E	F	G	H
Value	+2	+1	+3	+1	+2	+4	+2	+1

Arrival	1	2	1.2	1.8	1.6	1.4	2.2	1.6
1	<precomb> 3:FIRST							
1.2	<precomb> 1:FIRST		<precomb> 4:FIRST					
1.4	<precomb> 0:ROOT		<precomb> 1:SECOND (lock)			<precomb> 5:FIRST		
1.6	<comb> 3:FIRST, firstVal=2 (lock)		<comb> 4:FIRST firstVal=3		<precomb> 5:SECOND (lock)	<precomb> 2:FIRST		<precomb> 6:FIRST
1.8	<comb> 1:SECOND (wait)		<op> 1:SECOND sndVal = 3 (unlock,wait)	<precomb> 4:FIRST (wait)	<op> 5:SECOND sndVal=2 (unlock,wait)	<precomb> 0:ROOT		<precomb> 2:SECOND (lock)
2	<comb> 1:SECOND fstVal=2 sndVal=3	<precomb> 3:FIRST (wait)	(wait)	(wait)	(wait)	<comb> 5:SECOND firstVal=4 sndVal=2 (lock)		<comb> 6:FIRST, firstVal=1
2.2	<op> 0:ROOT result=5 returns 0	(wait)	(wait)	(wait)	(wait)	<comb> 2:SECOND (wait)	<precomb> 6:FIRST (wait)	<op> 2:SECOND sndVal=1 (unlock,wait)
2.4	<dist> 1:RESULT fstVal=2 sndVal=3 result=2	(wait)	(wait)	(wait)	(wait)	<comb> 2:SECOND firstVal=6 sndVal=1	(wait)	(wait)
2.6	<dist> 3:IDLE firstVal=2 (unlock)	(wait)	<op> 1:IDLE firstVal=2 sndVal=3 result=2 returns 2	(wait)	(wait)	<op> 0:ROOT result=12 returns 5	(wait)	(wait)
2.8	**답: 0**	<precomb> 3:FIRST	<dist> 4:IDLE (unlock)	(wait)	(wait)	<dist> 2:RESULT firstVal=6 sndVal=1 result=11	(wait)	(wait)
3		<precomb> 1:FIRST	**답:2**	<precomb> 4:FIRST	(wait)	<dist> 5:RESULT firstVal=4 sndVal=2 result=9	(wait)	<op> 2:IDLE result=11
3.2		<precomb> 0:ROOT		<precomb> 1:SECOND (lock)	<op> 5:IDLE result=9	**답:5**	(wait)	<dist> 6:IDLE
3.4		<comb> 3:FIRST firstValue=1		<comb> 4:FIRST firstValue=1	**답:9**		<precomb> 6:FIRST	**답:11**
3.6		<comb> 1:SECOND (wait)		<op> 1:RESULT sndValue=1 (unlock,wait)			<precomb> 2:FIRST	
3.8		<comb> 1:SECOND		(wait)			<precomb> 0:ROOT	

		fstValue=1 sndValue=1						
4		<op> 0:ROOT result=14 return 12		(wait)			<comb> 6:FIRST firstVal=2	
4.2		<dist> 1:RESULT fstValue=1 result=13		(wait)			<comb> 2:FIRST firstVal=2	
4.4		<dist> 3:IDLE		<op> 1:IDLE result=13			<op> 0:ROOT result=16 return 14	
4.6		**답:12**		<dist> 4:IDLE			<dist> 2:IDLE	
4.8				**답:13**			<dist> 6:IDLE	
5							**답:14**	

6. Tree-based bounded P-Q

Thread	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
Priority	1	4	Remove	0	4	7	Remove	2	0	Remove
Arrival	0	0.1	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.4
0	D0(0)									
0.1	B+(1)	F+(1)								
0.2	A+(1)	C+(1)	A-(0)	D+(1)	F+(2)	G0(0)				
0.3		A0(0)	B-(0)	B+(1)	C+(2)	C0(2)	A0(0)	E+(1)	D+(2)	
0.4			D-(1)	A+(1)	A0(1)	A0(1)	C-(1)	B0(1)	B+(2)	A-(0)
0.5			Remove T4(0)				F-(1)	A+(1)	A+(2)	B-(1)
0.6							Remove T2(4)			D-(0)
0.7										Remove T9(0)
0.8										

T4 – T3 – T2 – T7 – T1 – T5 – T6 – T8 – T9 – T10