Lösningsförslag till tentamen * *Preliminär*

Kursnamn	**Algoritmer och datastrukturer, 4p**
Tentamensdatum	**2000-08-24**
Program	**DAI 2**
Läsår	**99/00, lp I/II**
Examinator	**Uno Holmer**

*** se även `http://www.chl.chalmers.se/~holmer/` där finns det mesta av kursmaterialet.**

Uppgift 1 (10 p)

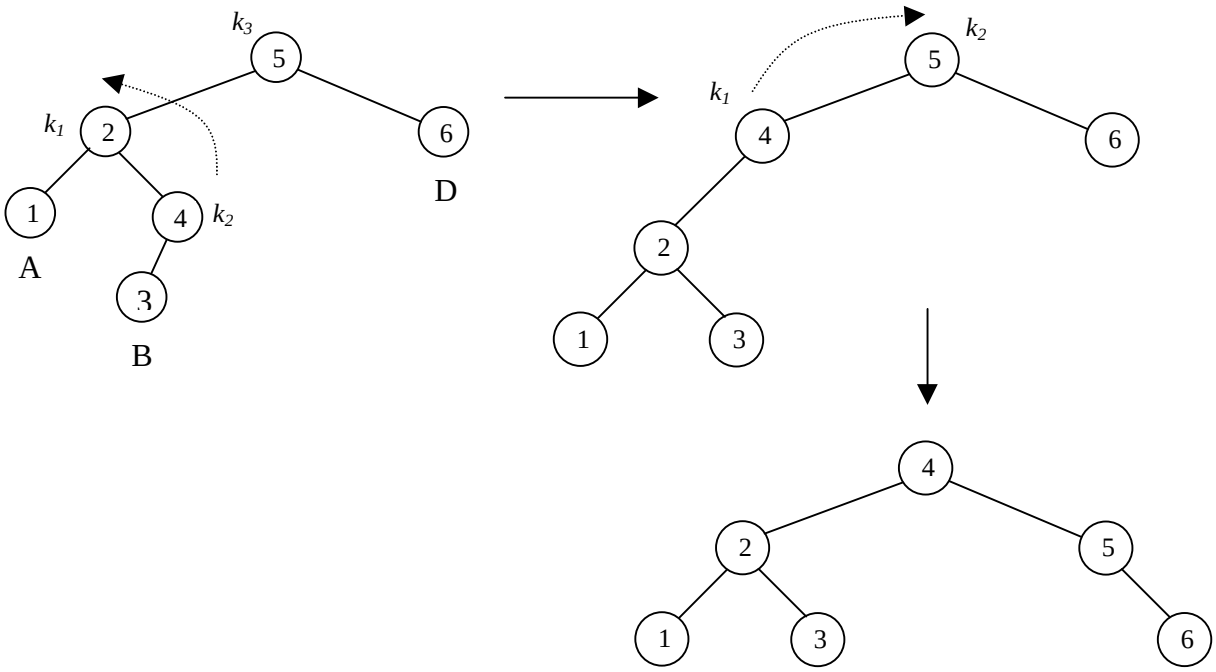
Ingen lösning ges

Uppgift 2 (8 p)

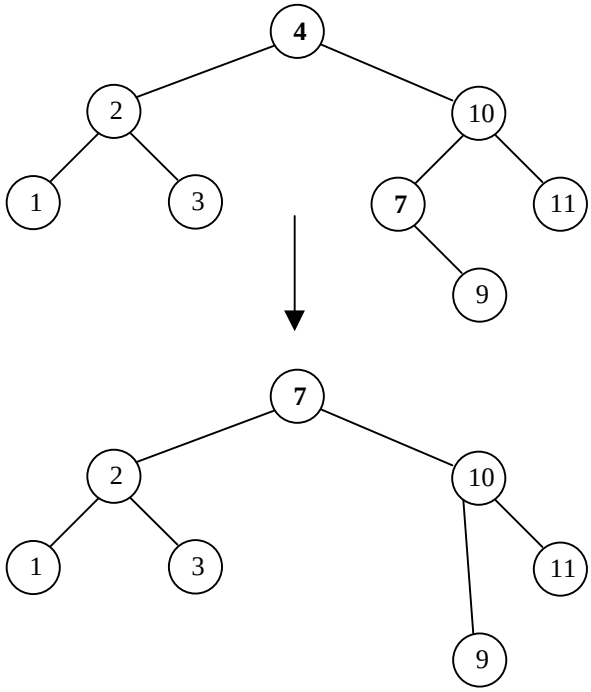
```
bool
Mobile::Equivalent ( const Mobile & Rhs ) const {
    const float eps = 0.000001;
    return Simple() && Rhs.Simple() &&
           fabs( Mass() - Rhs.Mass() ) < eps ||
           ! ( Simple() || Rhs.Simple() ) &&
           ( Left->Equivalent( *(Rhs.Left) ) &&
             Right->Equivalent( *(Rhs.Right) )&&
             fabs( LeftLength - Rhs.LeftLength ) < eps &&
             fabs( RightLength - Rhs.RightLength ) < eps  ||

             Left->Equivalent( *(Rhs.Right) ) &&
             Right->Equivalent( *(Rhs.Left) ) &&
             fabs( LeftLength - Rhs.RightLength ) < eps &&
             fabs( RightLength - Rhs.LeftLength ) < eps
           );
}
```

Uppgift 3 (6 p)

a) (3 p)
Detta motsvarar fall 2 i Weiss och balanseras med en vänster-höger-rotation.
Det sist insatta talet var 3.



b) (3 p)
Då Remove appliceras på en nod med två barn (4) ersätts noden med det minsta elementet (7) i det högra delträdet, vilket kan ha högst ett barn, varefter noden som innehöll detta tas bort:



Uppgift 4 (6 p)

Sekvensen Insert(24); Insert(49); ger följande resultat

a) (3 p) om $\lambda > 0.5$ tillåtet

0	**24**	$24 + 2^2 \mod 7 = 0$
1	*15*	
2	**49**	$49 + 3^2 \mod 7 = 2$
3	*31*	
4	*17*	
5		
6		

b) (3 p) om $\lambda \leq 0.5$ krävs

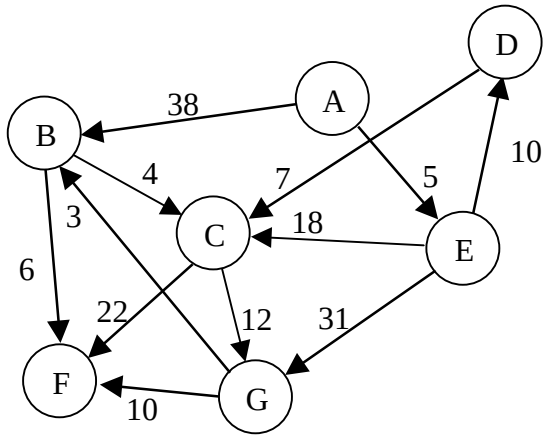
0	*17*	17 mod 17 = 0
1		
2		
3		
4		
5		
6		
7	**24**	24 mod 17 = 7
8		
9		
10		
11		
12		
13		
14	*31*	31 mod 17 = 14
15	*15*	15 mod 17 = 15
16	**49**	$49 + 1^2 \mod 17 = 16$

Observera att storleken på tabellerna är primtal. 17 är det minsta primtalet ≥ 2*7.

Uppgift 5 (9 p)

a) (3 p)

	A	B	C	D	E	F	G
A	0	38	∞	∞	5	∞	∞
B	∞	0	4	∞	∞	6	∞
C	∞	∞	0	∞	∞	22	12
D	∞	∞	7	0	∞	∞	∞
E	∞	∞	18	10	0	∞	31
F	∞	∞	∞	∞	∞	0	∞
G	∞	3	∞	∞	∞	10	0



b)

b) (3 p)

Kortaste vägarna från A till övriga noder, samt kostnaden för resp väg inom parentes är

	B	**C**	**D**	**E**	**F**	**G**
*** oviktad**	AB (1)	ABC (2)	AED (2)	AE (1)	ABF (2)	AEG (2)
viktad	AEDCGB (37)	AEDC (22)	AED (15)	AE (5)	AEDCGBF (43)	AEDCG (34)

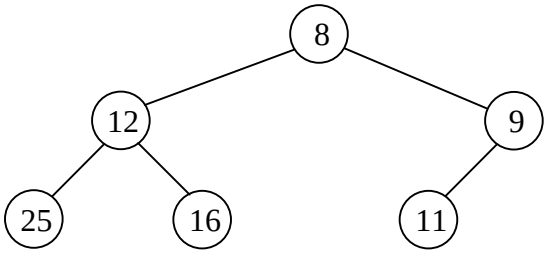
*) I vissa fall finns alternativa vägar av samma längd.

c) (3 p)

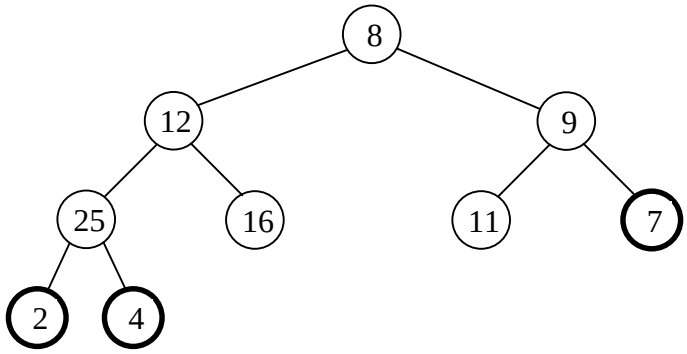
Grafen innehåller cykeln BCGB och det finns därför ingen topologisk ordning som innehåller samtliga noder.

Uppgift 6 (6 p)

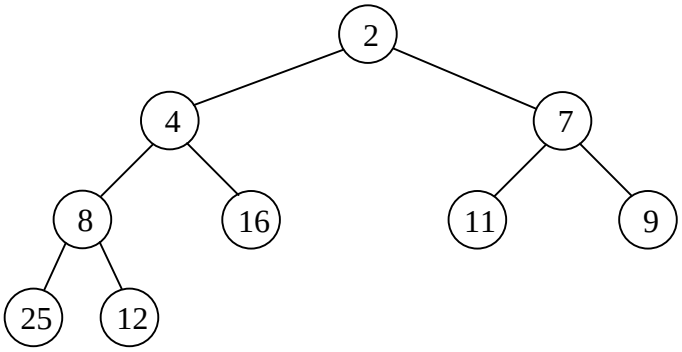
a) (1 p) *H* som träd



b) (1 p) Efter Toss(7); Toss(2); Toss(4);



c) (4 p) Efter FixHeap()



Uppgift 7

a) (5 p)

```
PrintJob::operator< ( const PrintJob & Rhs ) const {
    return _Size < Rhs._Size ||
              ( _Size == Rhs._Size &&
                  ( _ArrivalTime < Rhs._ArrivalTime ||
                      ( _ArrivalTime == Rhs._ArrivalTime &&
                          _Id < Rhs._Id ) ) );
}
```

b) (10 p)

```
void Simulator::Simulate( unsigned int SimulationTime ) {
    unsigned int NextId = 1;
    unsigned int TimeForNextJob = 0;
    unsigned int BusyTime = 0;
    BinaryHeap<PrintJob> TheQueue( PrintJob() );

    for ( unsigned int T = 0; T <= SimulationTime; T++ ) {
        PrintJob J;
        if ( BusyTime == 0 && ! TheQueue.IsEmpty() ) {
            // Service the first job in queue
            TheQueue.DeleteMin( J );
            J.Print();
            BusyTime = J.Size()/256;
        }
        if ( T == TimeForNextJob ) {
            // Enqueue a new random job
            PrintJob J( NextId++,
                        1 + random( 100000 ),  // Random size
                        T ); // Time of arrival to queue
            TheQueue.Insert( J );
            // Random time of next job insertion
            TimeForNextJob += 1 + random( 600 );
        }
        if ( BusyTime > 0 )
            BusyTime--;
    }
}
```