Lösningsförslag till tentamen *

Kursnamn	**Algoritmer och datastrukturer, 4p**
Tentamensdatum	**2000-12-11**
Program	**DAI 2**
Läsår	**2000/2001, 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. Se kurslitteraturen.

Uppgift 2 (4+6 p)

a) (4 p)

```
Node *Copy( const Node *L ) {
    if ( L == NULL )
        return NULL;
    else
        return new Node( L->Data, Copy( L->Next ) );}
```

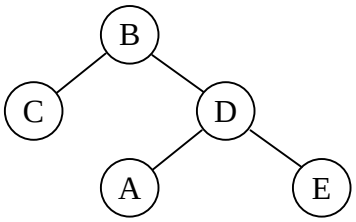
b) (6 p)

```
Node *Append( const Node *L1, const Node *L2 ) {
    if ( L1 == NULL )
        return Copy( L2 );
    else
        return new Node( L1->Data, Append( L1->Next, L2 ) );
}

// Alt. utan Copy:
Node *Append( const Node *L1, const Node *L2 ) {
    if ( L1 == NULL && L2 == NULL )
        return NULL;
    else if ( L1 == NULL )
        return Append( L2, NULL );
    else
        return new Node( L1->Data, Append( L1->Next, L2 ) );
}
```

Uppgift 3 (2+6 p)

a) (2 p)



b) (6 p)

```
inline int Max( int X, int Y ) { return X > Y ? X : Y; }

int Height( const Node *T ) {
    if ( T == NULL )
        return -1;
    else
        return 1 + Max( Height( T->Left ), Height( T->Right ) );
}

bool IsAVL( const Node *T ) {
    return T == NULL ||
        ( IsAVL( T->Left ) && IsAVL( T->Right ) &&
          abs( Height( T->Left ) - Height( T->Right ) ) <= 1 );
}
```

Uppgift 4 (2+4 p)

a) (2 p)
Söktiden för lyckad sökning blir i genomsnitt *O*(N) eftersom alla elementen hamnar i samma kollisionslista.

b) (4 p)
Insert(112) kräver omhashning. Eftersom kvadratisk sondering används skall tabellstorleken vara ett primtal. Det minsta primtalet ≥ 2*7 är 17. Elementens placering i den nya tabellen beror på insättningsordningen. Nedan visas hur det kan se ut om de befintliga elementen sätts in i den nya tabellen innan 112 sätts in.

0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
10	78	78 mod 17 = 10
11	130	130 mod 17 = 11
12	28	$28 + 1^2 \bmod 17 = 12$
13		
14	112	$112 + 2^2 \bmod 17 = 14$
15		
16		

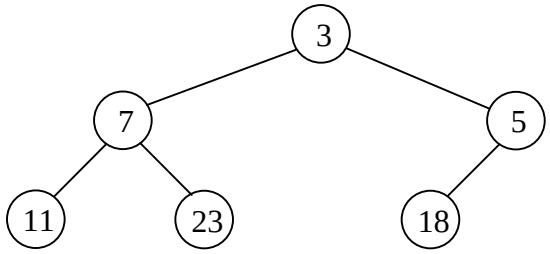
Uppgift 5 (2+4 p)

a) (2 p)
Ja, om grannmatrisen har den formen kan grafen ej innehålla några cykler och då finns minst en topologisk ordning som omfattar samtliga noder. Säg att grafens noder är numrerade 1,…,N. Då är samtliga bågar i grafen riktade från noder med lägre nummer till noder med högre. En båge riktad från en nod med högre nummer till en med lägre skulle motsvaras av ett tal under diagonalen i matrisen och några sådana finns ej enligt figuren. Enligt förutsättningarna finns det till och med bågar från varje nod till *alla* med högre nummer. För en sådan graf finns exakt en topologisk sortering av noderna, nämligen just 1,...,N.
Anm. Den andra extremen är en osammanhängande graf som helt saknar bågar (samtliga matriselement är ∞). Då finns inga restriktioner utan alla permutationer av 1,...,N är korrekta topologiska ordningar eftersom alla noder har ingrad 0.

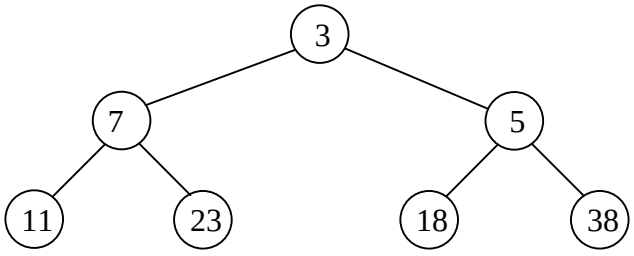
b) (4 p)
Om tryckeriet ligger i E tar hela distributionen högst tre timmar. Det behövs fyra bilar. Alla kör samtidigt från E, en vardera till A och C och två till D varav den ena kör direkt vidare till B. Den andra har en timma på sig att lasta av tidningar i D innan den fortsätter till F.

Uppgift 6 (1+4 p)

a) (1 p) *H* som träd



b) (4 p) Efter Insert(2); Insert(38); DeleteMin();



Uppgift 7 (15 p)

```
class WeatherReport {
 ...
private:
    ifstream In;
    SearchTree<SensorMinMax> theTree;
};

WeatherReport::WeatherReport( const char *File ) {
    In.open(File, ios::binary);
    if ( ! In ) {
        cerr << "Cannot open " << File << endl;
        exit(1);
    }
}

void WeatherReport::ComputeMinMax() {
    while ( true ) {
        static SensorData D;
        In.read( (char*)&D, sizeof(D) );
        if ( In.eof() )
            break;
        SensorMinMax SMM( D.Id );
        // Look up D in the Tree
        const SensorMinMax &OldMinMax = theTree.Find( SMM );
        // If not present, insert it
        if ( ! theTree.WasFound() )
            theTree.Insert(
                    SensorMinMax( D.Id,
                                  D.Temperature,
                                  D.Temperature ) );
        else {
            // If present, update the old one
            if ( D.Temperature < *OldMinMax.MinTemp )
                *OldMinMax.MinTemp = D.Temperature;
            if ( D.Temperature > *OldMinMax.MaxTemp )
                *OldMinMax.MaxTemp = D.Temperature;
        }
    }
}

void WeatherReport::ReportMinMax() {
    InOrder<SensorMinMax> Itr(theTree);
    if ( ! theTree.IsEmpty() )
        for( Itr.First(); +Itr; ++Itr ) {
            const SensorMinMax & X = Itr();
            cout << setw(10) << X.Id
                 << "  min: "
                 << setw(5) << *X.MinTemp
                 << "  max: "
                 << setw(5) << *X.MaxTemp << endl;
        }
}
```