Lösningsförslag till tentamen * *Preliminär*

Kursnamn	**Algoritmer och datastrukturer, 4p**
Tentamensdatum	**2001-04-18**
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 (10 p)

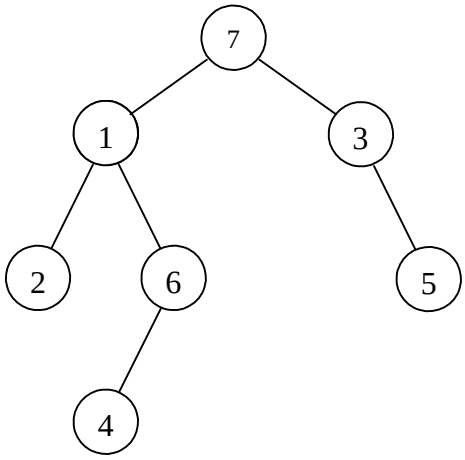
```
// Med copy
Node *Copy( const Node *L ) {
    if ( L == NULL )
        return NULL;
    else
        return new Node( L->Data, Copy( L->Next ) );
}

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

// Utan copy
Node *Merge( const Node *L1, const Node *L2 ) {
    if ( L1 == NULL && L2 == NULL )
        return NULL;
    else if ( L1 == NULL )
        return new Node( L2->Data, Merge( NULL, L2->Next ) );
    else if ( L2 == NULL )
        return new Node( L1->Data, Merge( L1->Next, NULL ) );
    else if ( L1->Data < L2->Data )
        return new Node( L1->Data, Merge( L1->Next, L2 ) );
    else
        return new Node( L2->Data, Merge( L1, L2->Next ) );
}
```

Uppgift 3 (2+4+6 p)

a) (2 p)



b) (4 p)

```
int Max( const Tree *T ) {
   if ( T == NULL )
      return INT_MIN;
   else
      return max( Max( T->Left ),
                  max( T->Data, Max( T->Right ) ) );
}
```

c) (6 p)

```
bool IsOrdered( const Tree *T ) {
   return T == NULL ||
          ( IsOrdered( T->Left ) &&
            IsOrdered( T->Right ) &&
            Max( T->Left ) <= T->Data &&
            Min( T->Right ) >= T->Data );
}
```

Uppgift 4 (3+4 p)

a) (3 p)

0	
1	
2	20
3	12
4	3
5	101
6	
7	
8	

b) (4 p)

Antag att den tomma tabellen har 11 positioner från början. Hashvärdena för talen blir 24→2, 47→3, 72→6, 13→2, 2→2. Först kan talen 24, 47 och 72 ha satts in i godtycklig ordning och utan kollisioner, därefter måste 13 satts in följt av 2. Inget av 13 eller 2 kan ha satts in före de tre övriga eftersom de talen då skulle hamnat i andra positioner p.g.a. kollisioner. 13 måste ha satts in före 2. Första lediga plats för 13 var $(2+3^2)$ mod 11 = 0, och för 2 $(2+4^2)$ mod 11 = 7. Om 2 och satts in före 13 skulle de bytt plats med varandra.

Uppgift 5 (3+3 p)

a) (3 p) ADBCE, ABDCE, ABCDE, BACDE, BADCE, BCADE.

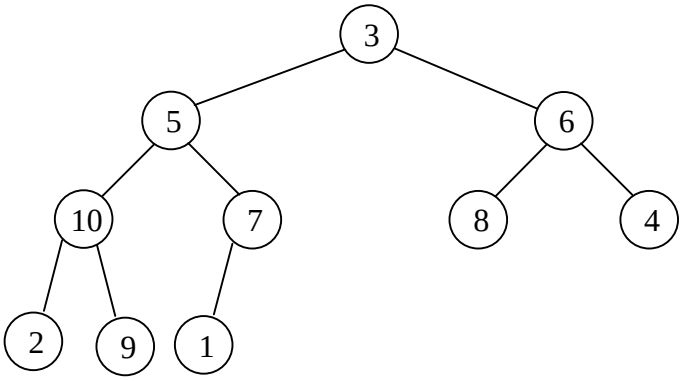
b) (3 p)

	B	**C**	**D**	**E**	**F**	**G**
oviktad	AB	AC	AD	ACE	AF	ACG
viktad	AFDCB(48)	AFDC(36)	AFD(25)	AFDCBE(62)	AF(12)	AFDG(68)

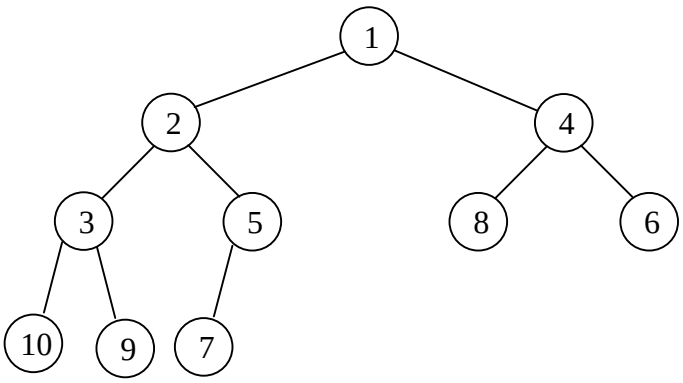
För de oviktade vägarna finns i vissa fall alternativ.

Uppgift 6 (1+4 p)

a) (1 p)



b) (4 p)



Uppgift 7 (Leta tecken) (10 p)

```
Set MkSet( const unsigned char *Str ) {
    Set S( 256 );
    while ( *Str )
        S += *Str++;
    return S;
}

void main() {
    Set All( 256 );
    unsigned char Buf[ 256 ];
    cout << "Skriv några rader, avsluta med CTRL-Z" << endl;

    // Get the first line
    cin.getline( Buf, 256 );
    if ( ! cin.eof() ) {
        // Initialize the set with chars from the first line
        All = MkSet( Buf );
        while ( true ) {
            cin.getline( Buf, 256 );
            if ( cin.eof() )
               break;
            All = All && MkSet( Buf ); // Intersection
        }
    }
    cout << "Gemensamma tecken: " << endl;
    for ( int C = 32; C < 256; C++ )
       if ( C < All )
          cout << (unsigned char)C; // important type conversion
    cout << endl;
}
```