

**Opgave 1 (4%)**

	Ja	Nej
$n + n^2$ er $O(n^2)$ ?	<input checked="" type="checkbox"/>	<input type="checkbox"/>
$n^3 \cdot \log n$ er $O(n^3)$ ?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
$n^3 + \log n$ er $O(n^3)$ ?	<input checked="" type="checkbox"/>	<input type="checkbox"/>
$2^{\log n}$ er $O(n^3)$ ?	<input checked="" type="checkbox"/>	<input type="checkbox"/>
$3^n$ er $O(\sqrt{n})$ ?	<input type="checkbox"/>	<input checked="" type="checkbox"/>

**Opgave 2 (4%)**

Opskriv følgende funktioner efter stigende orden med hensyn til  $O$ -notationen:

$$4n^2$$
$$2^{3 \log n}$$
$$2^n$$
$$1 / \log n$$
$$\sqrt{n} \cdot \log n$$

Svar: \_\_\_\_\_  $1 / \log n$      $\sqrt{n} \cdot \log n$      $4n^2$      $2^{3 \log n}$      $2^n$

**Opgave 3 (4%)**

I det følgende angiver  $f_i$  og  $g_i$  positive ikke-aftagende funktioner, hvor  $f_1(n) = O(g_1(n))$  og  $f_2(n) = O(g_2(n))$ . Angiv hvilke af følgende udsagn der er sande.

	Ja	Nej
$f_1(n) + f_2(n) = O(g_1(n) + g_2(n))$ ?	<input checked="" type="checkbox"/>	<input type="checkbox"/>
$f_1(n) - f_2(n) = O(g_1(n) - g_2(n))$ ?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
$f_1(n) * f_2(n) = O(g_1(n) * g_2(n))$ ?	<input checked="" type="checkbox"/>	<input type="checkbox"/>
$f_1(n) / f_2(n) = O(g_1(n) / g_2(n))$ ?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
$1 / f_1(n) = \Omega(1 / g_1(n))$ ?	<input checked="" type="checkbox"/>	<input type="checkbox"/>

**Opgave 4 (4%)**

Angiv for hver af nedenstående algoritmer udførelstiden som funktion af  $n$  i  $O$ -notation.

**Algoritme Loop1( $n$ )**

```
s = 1
for i = 1 to n
  for j = 1 to n
    s = s + 1
```

**Algoritme Loop2( $n$ )**

```
s = 1
for i = 1 to n
  for j = i to n
    for k = i to j
      s = s + 1
```

**Algoritme Loop3( $n$ )**

```
for i = 1 to n
  j = 1
  while j ≤ i
    j = 2 * j
```

Svar Loop1: \_\_\_\_\_  $O(n^2)$  \_\_\_\_\_

Svar Loop2: \_\_\_\_\_  $O(n^3)$  \_\_\_\_\_

Svar Loop3: \_\_\_\_\_  $O(n \log n)$  \_\_\_\_\_

**Opgave 5 (4%)**

Angiv for hver af nedenstående algoritmer udførelstiden som funktion af  $n$  i  $O$ -notation.

**Algoritme Loop1( $n$ )**

```
i = 1
while i ≤ n
  j = 1
  while j ≤ i
    j = j + 1
  i = 2 * i
```

**Algoritme Loop2( $n$ )**

```
i = 1
while i ≤ n
  i = i + i
```

**Algoritme Loop3( $n$ )**

```
i = 2
while i ≤ n
  i = i * i
```

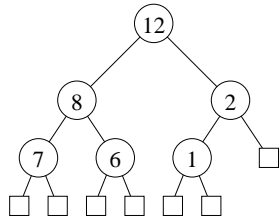
Svar Loop1: \_\_\_\_\_  $O(n)$  \_\_\_\_\_

Svar Loop2: \_\_\_\_\_  $O(\log n)$  \_\_\_\_\_

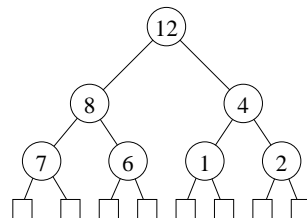
Svar Loop3: \_\_\_\_\_  $O(\log \log n)$  \_\_\_\_\_

**Opgave 6 (4%)**

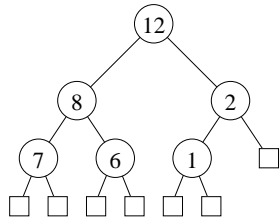
Tegn hvordan nedenstående binære max-heap ser ud efter indsættelse af elementet 4.



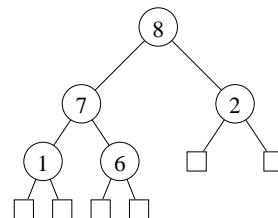
Svar: \_\_\_\_\_



Tegn hvordan nedenstående binære max-heap ser ud efter en heap-extract-max operation.



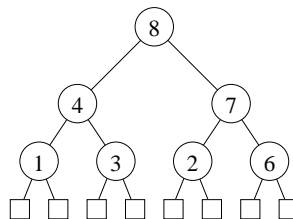
Svar: \_\_\_\_\_



**Opgave 7 (4%)**

Tegn den binære max-heap efter indsættelse af elementerne 1, 3, 2, 4, 6, 8, 7 i den givne rækkefølge, startende med den tomme heap.

Svar: \_\_\_\_\_



**Opgave 8 (4%)**

Angiv hvordan nedenstående array ser ud efter anvendelsen af build-max-heap for arrayet.

1	2	3	4	5	6	7	8	9	10
4	2	5	1	10	7	8	3	6	9

1	2	3	4	5	6	7	8	9	10
10	9	8	6	4	7	5	3	1	2

Svar: \_\_\_\_\_

**Opgave 9 (4%)**

Betragt radix-sort anvendt på nedenstående liste af binære tal ( $d = 6, k = 2$ ). Angiv den delvist sorterede liste efter at radix-sort har sorteret tallene efter de *tre* mindst betydende cifre.

110101    110001    000101    000001    101101    001001

Svar: 110001    000001    001001    110101    000101    101101

**Opgave 10 (4%)**

Angiv resultatet af at anvende PARTITION( $A, 11, 19$ ) på nedenstående array.

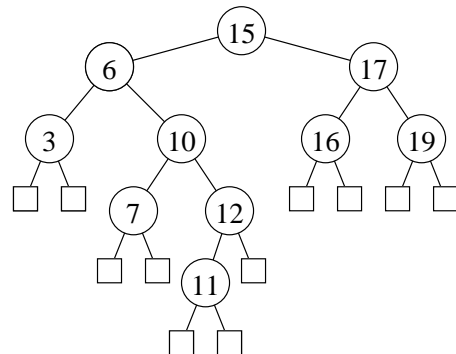
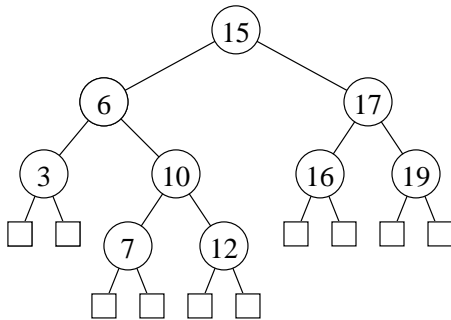
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
A	8	16	1	6	2	4	13	17	15	3	18	5	9	11	24	12	14	10	7	22

Svar: 

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
	8	16	1	6	2	4	13	17	15	3	5	7	9	11	24	12	14	10	18	22

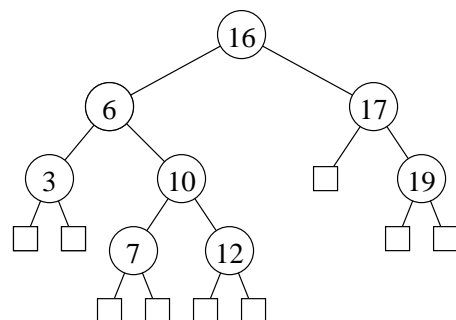
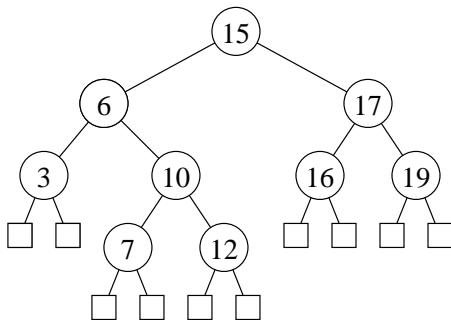
**Opgave 11 (4%)**

Tegn hvordan nedenstående ubalancerede binære søgetræ ser ud efter indsættelse af elementet 11.



Svar: \_\_\_\_\_

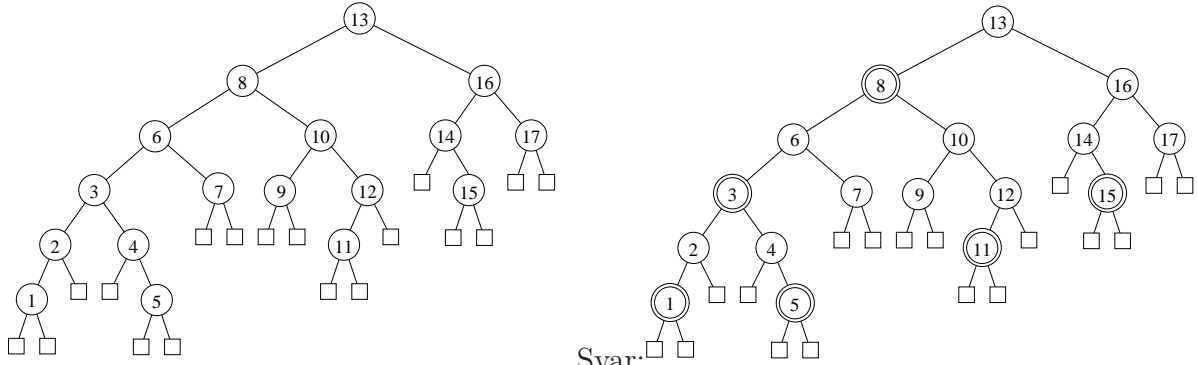
Tegn hvordan nedenstående ubalancerede binære søgetræ ser ud efter slettelse af elementet 15.



Svar: \_\_\_\_\_

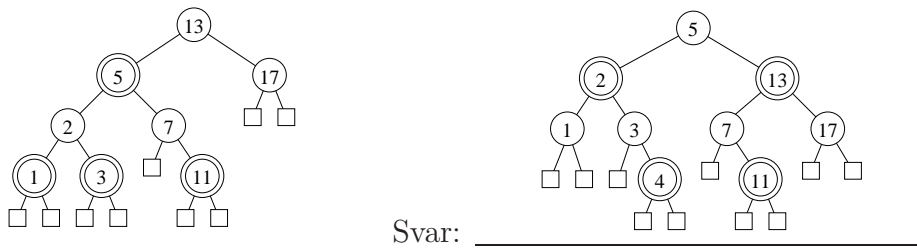
**Opgave 12 (4%)**

Angiv hvorledes knuderne i nedenstående binære søgetræ kan farves røde og sorte, således at det resulterende træ er et lovligt rød-sort træ.



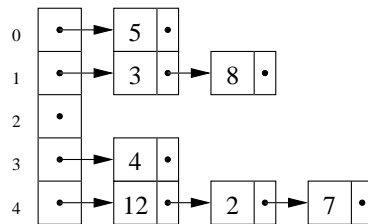
**Opgave 13 (4%)**

Tegn hvordan nedenstående rød-sort træ (dobbeltcirkler angiver røde knuder) ser ud efter indsættelse af elementet 4.



**Opgave 14 (4%)**

Tegn en hashtabel hvor der anvendes kædede lister til at håndtere kollisioner, når hash-funktionen er  $h(k) = 2k \bmod 5$  og der indsættes elementerne 5, 7, 2, 4, 8, 3, og 12 i den givne rækkefølge.



Svar: \_\_\_\_\_

**Opgave 15 (4%)**

Tegn hvordan en hashtabel der anvender *linear probing* ser ud efter at elementerne 2, 6, 12, 1, 5, 10, 16, og 0 indsættes i den givne rækkefølge, når hashfunktionen er  $h(k) = 2k \bmod 11$ .

0	1	2	3	4	5	6	7	8	9	10

0	1	2	3	4	5	6	7	8	9	10
16	6	12	1	2	0				10	5

Svar: \_\_\_\_\_

**Opgave 16 (4%)**

Tegn hvordan en hashtabel der anvender *dobbelt hashing* ser ud efter at elementerne 11, 4, 3, 5, og 12 indsættes i den givne rækkefølge, når hashfunktionerne er  $h_1(k) = k \bmod 8$  og  $h_2(k) = 1 + 2k \bmod 8$ , og hashtabellen har størrelse 8.

0	1	2	3	4	5	6	7

0	1	2	3	4	5	6	7
		3	11	4	5	12	

Svar: \_\_\_\_\_

**Opgave 17 (4%)**

Angiv for hver af nedenstående datastrukturer indeholdende  $n$  elementer, hvor lang tid det tager at finde predecessoren til et element  $x$ , dvs. det største element  $\leq x$  i datastrukturen, som funktion af  $n$  i  $O$ -notation.

Rød-sort søgetræ ?

Svar:            $O(\log n)$           

Binær max-heap ?

Svar:            $O(n)$           

Hashtabel med linear probing ved 50% fyldningsgrad ?

Svar:            $O(n)$



<b>Transitionssystem</b> Frem-og-tilbage Konfigurationer: $\{[i, j, k] \mid \text{heltal } i, j, k \wedge i \geq 0 \wedge j \geq 0 \wedge k \geq 0\}$ $[i, j, k] \triangleright [i - 1, j + 2, k] \quad \text{if } i \geq 1$ $[i, j, k] \triangleright [i, j - 1, k + 3] \quad \text{if } j \geq 1$ $[i, j, k] \triangleright [i + 1, j, k - 7] \quad \text{if } k \geq 7$
--

**Opgave 20 (4%)**

For hvert af nedenstående udsagn, angiv om de er en invariant for ovenstående transitionssystem Frem-og-tilbage. Startkonfigurationen antages at være  $[n, n, n]$  hvor  $n \geq 0$ .

	Ja	Nej
$i + j + k \geq 0$	<input checked="" type="checkbox"/>	<input type="checkbox"/>
$i \leq j$	<input type="checkbox"/>	<input checked="" type="checkbox"/>
$6i + 3j + k \leq 10n$	<input checked="" type="checkbox"/>	<input type="checkbox"/>
$2i + j \leq 3k$	<input type="checkbox"/>	<input checked="" type="checkbox"/>
$i - 1 = j + 2$	<input type="checkbox"/>	<input checked="" type="checkbox"/>

**Opgave 21 (4%)**

For hver af nedenstående funktioner, angiv om de er en termineringsfunktion for ovenstående transitionssystem Frem-og-tilbage.

	Ja	Nej
$\mu(i, j, k) = i + j + k$	<input type="checkbox"/>	<input checked="" type="checkbox"/>
$\mu(i, j, k) = k$	<input type="checkbox"/>	<input checked="" type="checkbox"/>
$\mu(i, j, k) = 52i + 25j + 8k$	<input checked="" type="checkbox"/>	<input type="checkbox"/>
$\mu(i, j, k) = 10i + 4j + k$	<input type="checkbox"/>	<input checked="" type="checkbox"/>
$\mu(i, j, k) = 6i + 3j + k$	<input type="checkbox"/>	<input checked="" type="checkbox"/>



**Algoritme** Factorial( $n$ )

Inputbetingelse : heltal  $n \geq 1$

Outputkrav :  $r = n! = 1 \cdot 2 \cdot 3 \cdots (n - 1) \cdot n$

Metode :  $i \leftarrow n$

$r \leftarrow 1$

$\{I\}$  **while**  $i > 1$  **do**

$r \leftarrow r * i$

$i \leftarrow i - 1$

**Opgave 22** (4%)

For hvert af nedenstående udsagn, angiv om de er en invariant  $I$  for ovenstående algoritme Factorial.

	Ja	Nej
$i \geq 1$	<input checked="" type="checkbox"/>	<input type="checkbox"/>
$r = i!$	<input type="checkbox"/>	<input checked="" type="checkbox"/>
$i! \cdot r! = n!$	<input type="checkbox"/>	<input checked="" type="checkbox"/>
$r = n!/i!$	<input checked="" type="checkbox"/>	<input type="checkbox"/>
$r = n!$	<input type="checkbox"/>	<input checked="" type="checkbox"/>

**Opgave 23** (4%)

For hver af nedenstående funktioner, angiv om de er en termineringsfunktion for ovenstående algoritme Factorial.

	Ja	Nej
$\mu(i, r) = i$	<input checked="" type="checkbox"/>	<input type="checkbox"/>
$\mu(i, r) = n! - r$	<input checked="" type="checkbox"/>	<input type="checkbox"/>
$\mu(i, r) = (n - i)!$	<input type="checkbox"/>	<input checked="" type="checkbox"/>
$\mu(i, r) = 2^i$	<input checked="" type="checkbox"/>	<input type="checkbox"/>
$\mu(i, r) = n!/r$	<input checked="" type="checkbox"/>	<input type="checkbox"/>

**Opgave 24 (4%)**

Givet et positivt heltal  $n$ , beregner nedenstående algoritme heltalslogarithmen af  $n$ . Denne værdi betegnes

$$\log(n) = \max\{i \mid i \text{ heltal} \wedge 2^i \leq n\} .$$

For at vise gyldigheden af algoritmen skal  $I_p$  og  $I_r$  være invarianter omkring variableerne  $p$  og  $r$ . Angiv invarianter hvormed gyldigheden af algoritmen kan bevises (bevis for invarianterne kræves ikke).

```
Algoritme IntegerLog( $n$ )  
Inputbetingelse : positivt heltal  $n$   
Outputkrav      :  $r = \log(n)$   
Metode          :  $p \leftarrow 1$ ;  
                  $r \leftarrow 0$ ;  
                  $\{I_p \wedge I_r\}$  while  $2 * p \leq n$  do  
                    $p \leftarrow 2 * p$ ;  
                    $r \leftarrow r + 1$ ;
```

Svar  $I_p$ : \_\_\_\_\_  $1 \leq p \leq n$  \_\_\_\_\_

Svar  $I_r$ : \_\_\_\_\_  $p = 2^r$  \_\_\_\_\_

For at kunne bevise at algoritmen terminerer, kræves en passende termineringsfunktion. Angiv en termineringsfunktion (bevis for at termineringsfunktionen har de nødvendige egenskaber kræves ikke).

Svar  $\mu$ : \_\_\_\_\_  $\log(n) - r$  \_\_\_\_\_

**Opgave 25 (4%)**

Antag en binær tæller implementeres som et (uendeligt) array af bits  $B[0]B[1]B[2] \dots$ . Det antages at tælleren forøges med én vha. følgende metode.

```
Algoritme inc  
Metode :  $i \leftarrow 0$ ;  
         while  $B[i] = 1$  do  
            $B[i] \leftarrow 0$ ;  
            $i \leftarrow i + 1$ ;  
          $B[i] \leftarrow 1$ 
```

For at argumentere at inc tager amortiseret  $O(1)$  tid kræves en potentiale funktion. Angiv en potentiale funktion hvorved tiden kan bevises. Argumentation for tiden kræves ikke.

Svar  $\Phi(B[0]B[1]B[2] \dots)$  : \_\_\_\_\_  $|\{i \mid B[i] = 1\}|$  \_\_\_\_\_