EKSAMEN

Algoritmer og Datastrukturer

Torsdag 27. maj 2021, 9:00–11:00

Institut for Datalogi, Naturvidenskabelige Fakultet, Aarhus Universitet

Antal sider i opgavesættet (incl. forsiden): 14

Tilladte medbragte hjælpemidler:
Alle, inklusive internet.
Det er ikke tilladt at kommunikere med andre under eksamen.

Studienummer :

Navn :

Dokumentet skal udfyldes med Adobe Acrobat,
som kan hentes fra `https://get.adobe.com/reader/`

Vejledning og pointgivning

Dette eksamenssæt består af en mængde multiple-choice-opgaver.

Opgaverne besvares på opgaveformuleringen **som afleveres**.

For hver opgave er angivet opgavens andel af det samlede eksamenssæt.

Hvert delspørgsmål har præcist et rigtigt svar.

For hvert delspørgsmål må du vælge **max ét svar** ved at afkrydse den tilsvarende rubrik.

Et delspørgsmål bedømmes som følgende:

- Hvis du sætter kryds ved det rigtige svar, får du 1 point.
- Hvis du ikke sætter nogen krydser, får du 0 point.
- Hvis du sætter kryds ved et forkert svar, får du $-\frac{1}{k-1}$ point, hvor k er antal svarmuligheder.

For en opgave med vægt $v\,\%$ og med n delspørgsmål, hvor du opnår samlet s point, beregnes din besvarelse af opgaven som:

$$\frac{s}{n} \cdot v\,\%$$

Bemærk at det er muligt at få negative point for en opgave.

Opgave 1 (Asymptotisk notation, 6 %) check

I det følgende angiver $\log n$ 2-tals-logaritmen af n.

	Ja	Nej
$(\log n)^2$ er $O(n^2)$	☒	☐
$n \cdot \log n$ er $O(n^2)$	☒	☐
$\sqrt{n}$ er $O((\log n)^3)$	☐	☒
$1 + \log(n^2)$ er $O((\log n)^2)$	☒	☐
$\log n + \log(n!)$ er $O(n^2)$	☒	☐
n^3 er $O(n)$	☐	☒
$\sqrt{n} \cdot \log n$ er $O(n)$	☒	☐
n^3 er $O(\log(n!))$	☐	☒
n^2 er $O(n^{2/3})$	☐	☒
$7 \log n + \log(n!)$ er $\Theta(n \cdot \log n)$	☐	☐
$2^{\log n}$ er $\Omega(n^{0.01})$	☐	☐
$(\log n)^3 + 3^n$ er $\Omega(2^n)$	☐	☐

Opgave 2 (Analyse af løkker, 6 %) check

Algoritme loop1(n)
```
s = 1
for i = 1 to n * n
    for j = 1 to n
        s = s + 1
```

Algoritme loop2(n)
```
i = n
while i ≤ n * n
    i = 2 * i
```

Algoritme loop3(n)
```
i = 1
j = n * n
while i ≤ j
    i = 2 * i
    j = j - 1
```

Algoritme loop4(n)
```
i = 1
while i ≤ n
    j = i
    while j > 0
        j = ⌊j/2⌋
    i = i + i
```

Angiv for hver af ovenstående algoritmer udførselstiden som funktion af n i Θ-notation.

	$\Theta(\sqrt{n})$	$\Theta(n^3)$	$\Theta(n)$	$\Theta(\sqrt[3]{n})$	$\Theta(n^2)$	$\Theta(\log n)$	$\Theta((\log n)^2)$	$\Theta(n \log n)$
loop1	☐	☒	☐	☐	☐	☐	☐	☐
loop2	☐	☐	☐	☐	☐	☒	☐	☐
loop3	☐	☐	☐	☐	☐	☒	☐	☐
loop4	☐	☐	☐	☐	☐	☐	☒	☐

Fra de hurtigste til de langsomste funktioner: log(n), \sqrt[3]{n}, √n, n, n*log(n), n^2, n^3

Opgave 3 (Max-Heap-Insert, 4 %)

check

Angiv den binære max-heap efter indsættelse af elementerne 6, 12, 13, 4, 8, 14 og 5 i den givne rækkefølge med MAX-HEAP-INSERT, startende med den tomme heap.

1	2	3	4	5	6	7	
14	12	13	4	8	6	5	☐
6	12	13	4	8	14	5	☐
14	8	13	4	6	12	5	☐
14	13	12	8	6	5	4	☐
13	12	14	4	8	6	5	☐

Opgave 4 (Heap-Extract-Max, 4 %)

check

1	2	3	4	5	6	7	8	9	10	11	12	13
25	24	20	13	23	16	15	1	9	4	3	12	14

Hvad er resultat af HEAP-EXTRACT-MAX på ovenstående max-heap ?

1	2	3	4	5	6	7	8	9	10	11	12	13	
24	23	20	13	4	16	15	1	9		3	12	14	☐
1	2	3	4	5	6	7	8	9	10	11	12		
24	23	20	13	4	16	15	1	9	3	12	14		☐
1	2	3	4	5	6	7	8	9	10	11	12		
24	23	20	13	4	16	15	1	9	14	3	12		☐
1	2	3	4	5	6	7	8	9	10	11	12		
24	23	20	13	14	16	15	1	9	4	3	12		☐
1	2	3	4	5	6	7	8	9	10	11	12		
24	23	15	20	16	14	1	9	4	3	12	13		☐

Opgave 5 (Sorteringsalgoritmer, 4 %)

check

Givet et array af størrelse n indeholdende tallene $1, 2, \ldots, n$ i voksende rækkefølge, hvad er worst-cast tiden for følgende sorteringsalgoritmer, når de anvendes på arrayet ?

	$\Theta(n)$	$\Theta(n \log n)$	$\Theta(n^2)$
InsertionSort	☒	☐	☐
HeapSort	☐	☒	☐
MergeSort	☐	☒	☐
QuickSort	☐	☐	☒

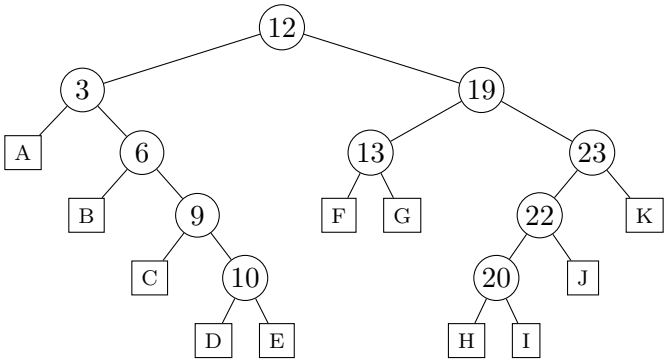
Opgave 6 (Partition, 4 %) check

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
21	17	28	14	9	18	6	1	26	15	30	7	13	19	2

Angiv resultatet af at anvende PARTITION$(A, 4, 13)$ på ovenstående array A.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
21	17	28	1	6	7	9	13	14	15	18	26	30	19	2	☐
1	2	6	7	9	13	14	15	17	18	19	21	26	28	30	☐
21	17	28	9	6	1	7	13	14	18	26	15	30	19	2	☐
21	17	28	9	6	1	7	13	26	15	30	14	18	19	2	☒

Opgave 7 (Indsættelser i søgetræer, 4 %) check

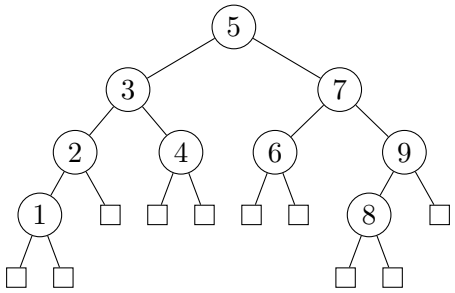


Angiv i hvilke blade A–K i ovenstående ubalancerede binære søgetræ elementerne 24, 18, 21, 17 og 8 skal indsættes (det antages at før hver indsættelse indeholder træet kun ovenstående ti elementer).

	A	B	C	D	E	F	G	H	I	J	K
INSERT(24)	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☒
INSERT(18)	☐	☐	☐	☐	☐	☐	☒	☐	☐	☐	☐
INSERT(21)	☐	☐	☐	☐	☐	☐	☐	☐	☒	☐	☐
INSERT(17)	☐	☐	☐	☐	☐	☐	☒	☐	☐	☐	☐
INSERT(8)	☐	☐	☒	☐	☐	☐	☐	☐	☐	☐	☐

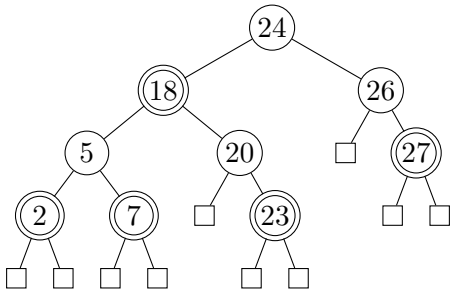
Opgave 8 (Rød-sort træ, 4 %) **check**

For hver af nedenstående delmængder, angiv om nedenstående binære træ er et lovligt rød-sort træ hvis netop disse knuder farves røde.

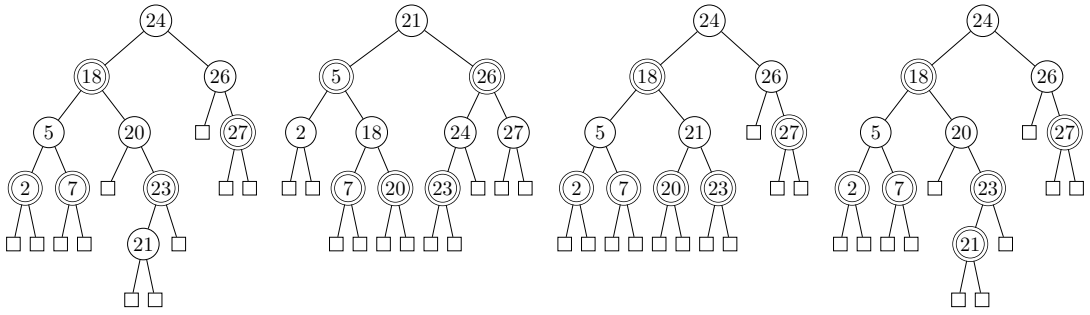


	Ja	Nej
1, 3, 6, 8, 9	☐	☒
1, 8	☒	☐
1, 2, 4, 7, 8	☐	☒
1, 5, 8	☐	☒
1, 3, 7, 8	☒	☐

Opgave 9 (Indsættelse i rød-sort træer, 4 %) **check**



Angiv det resulterende rød-sorte træ når man indsætter 21 i ovenstående rød-sorte træ (dobbeltcirkler angiver røde knuder).



☐ ☐ ☒ ☐

Opgave 10 (Lineær probing, 4 %) **check**

0	1	2	3	4	5	6	7	8	9	10
0			18		19	8				16

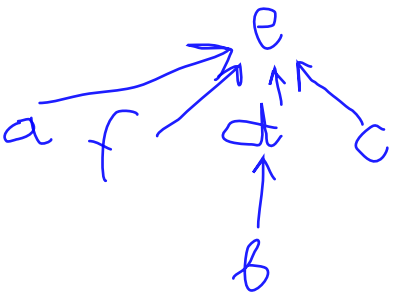
I ovenstående hashtabel af størrelse 11 er anvendt *linear probing* med hashfunktionen $h(k) = 2k \bmod 11$.

Angiv positionerne de fem elementer 2, 3, 5, 7 og 11 vil blive indsat på i hashtabellen (for hver af indsættelserne antager vi at hashtabellen kun indeholder elementerne 0, 8, 16, 18 og 19).

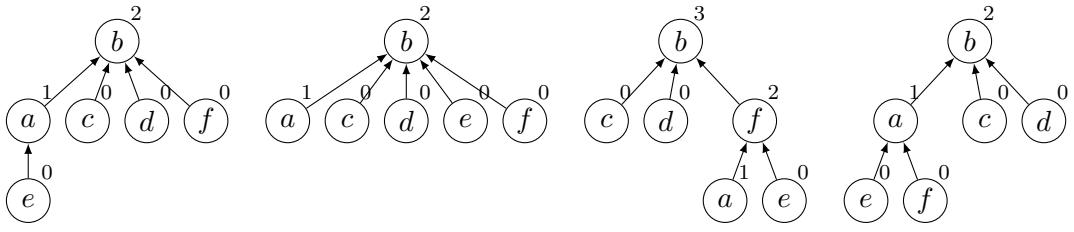
	0	1	2	3	4	5	6	7	8	9	10
INSERT(2)	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
INSERT(3)	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
INSERT(5)	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
INSERT(7)	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
INSERT(11)	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐

Opgave 11 (Union-find, 4 %) **check**

Angiv den resulterende union-find struktur efter nedenstående sekvens af operationer, når der anvendes union-by-rank og stikomprimering.

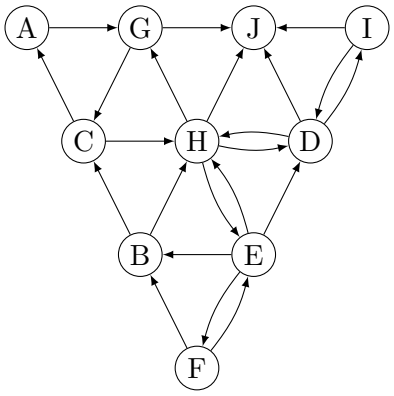


MAKESET(a)
MAKESET(b)
MAKESET(c)
MAKESET(d)
MAKESET(e)
MAKESET(f)
UNION(e, a)
UNION(a, f)
UNION(d, b)
UNION(e, b)
UNION(c, f)
FIND-SET(b)



☐ ☐ ☐ ☐

Opgave 12 (BFS, 4 %) **check**

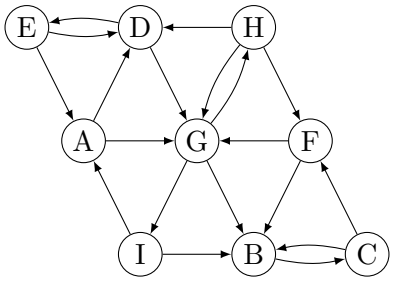


For et bredde først gennemløb (BFS) af ovenstående graf **startende i knuden A**, angiv rækkefølgen knuderne bliver indsat i køen Q i BFS-algoritmen. Det antages, at grafen er givet ved incidenslister, hvor incidenslisterne er sorteret alfabetisk.

A G C H D I J E B F A G C J H D E I F B A G J C H D E I F B A G C J H D E I B F

☐ ☐ ☐ ☐

Opgave 13 (DFS, 4 %) **check**



Betragt et dybde først gennemløb (DFS) af ovenstående graf, hvor DFS-gennemløbet starter i **knuden A**, hvor de udgående kanter til en knude besøges i alfabetisk rækkefølge. Angiv i hvilken rækkefølge knuderne får tildelt **discovery time**.

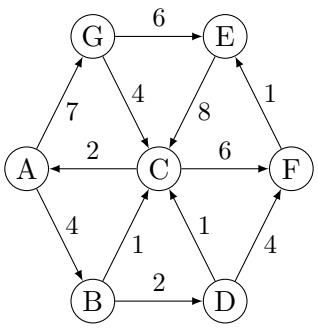
A G B C F H D E I A D E G B C F H I A D G E B H I C F A D G I H B C F E

☐ ☐ ☐ ☐

Angiv for hver af nedenstående kanter hvilken type kanten bliver i DFS gennemløbet.

	Tree edge	Back edge	Cross edge	Forward edge
(H, D)	☐	☐	☐	☐
(A, G)	☐	☐	☐	☐
(G, B)	☐	☐	☐	☐
(H, F)	☐	☐	☐	☐

Opgave 14 (Dijkstras algoritme, 4 %) check

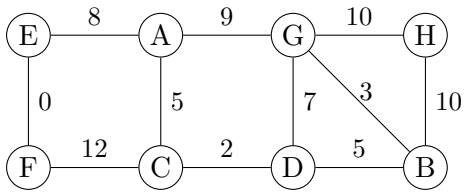


Antag Dijkstras algoritme anvendes til at finde korteste afstande fra **knuden A** til alle knuder i ovenstående graf. Angiv hvilken rækkefølge knuderne bliver taget ud af prioritetskøen i Dijkstra's algoritme.

A B C D G F E A B C D F E G A B G C D E F A B C F E D G

☒ ☐ ☐ ☐

Opgave 15 (Prims algoritme, 4 %) check



Antag Prims algoritme anvendes til at finde et minimum udspændende træ for ovenstående graf, og algoritmen starter i **knuden A**. Angiv hvilken rækkefølge knuderne bliver inkluderet i det minimum udspændende træ (taget ud af prioritetskøen i Prims algoritme).

A C D B G E F H A C D E F B G H A C D B G H F E A C D E F G B H

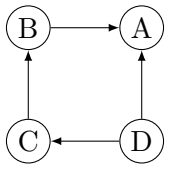
☒ ☐ ☐ ☐

Opgave 16 (Rekursionsligninger, 4 %) check

Angiv løsningen for hver af nedenstående rekursionsligninger, hvor $T(n) = 1$ for $n \leq 1$.

	$\Theta(\log n)$	$\Theta(\sqrt{n})$	$\Theta(n)$	$\Theta(n \log n)$	$\Theta(n^2)$	$\Theta(n^2 \log n)$	$\Theta(n^3)$
$T(n) = 4 \cdot T(n/2) + n^2$	☐	☐	☐	☐	☐	☐	☐
$T(n) = 2 \cdot T(n/5) + n$	☐	☐	☐	☐	☐	☐	☐
$T(n) = T(n-1) + \log n$	☐	☐	☐	☐	☐	☐	☐
$T(n) = T(n/4) + 5$	☐	☐	☐	☐	☐	☐	☐
$T(n) = 4 \cdot T(n/2) + 1$	☐	☐	☐	☐	☐	☐	☐

Opgave 17 (Topologisk sortering, 4 %) check



Angiv for hver af nedenstående ordninger af knuderne i ovenstående graf om det er en lovlig topologisk sortering.

	Ja	Nej
D C B A	☐	☐
B C D A	☐	☐
A C B D	☐	☐
D C A B	☐	☐
C D B A	☐	☐

Opgave 18 (Amortiseret analyse) check

En binær max-heap understøtter INSERT og HEAP-EXTRACT-MAX på en heap med n elementer i worst-case $O(\log n)$ tid. Bemærk den samme værdi kan være indsat flere gange i en max-heap. Vi ønsker nu at ændre HEAP-EXTRACT-MAX, således at den fjerner alle forekomster af maximum værdien fra heapen, d.v.s. den oprindelige HEAP-EXTRACT-MAX operation gentages indtil roden indeholder en mindre værdi eller at heapen er tom. Hvis maximum forekommer m gange i heapen, så vil HEAP-EXTRACT-MAX operationen tage worst-case $O(m \log n)$ tid.

Angiv for hver af nedenstående funktioner om de er en potentialefunktion, hvormed man kan argumentere for at operationerne INSERT og HEAP-EXTRACT-MAX tager amortiseret $O(\log n)$ tid. Antal elementer i heapen betegnes n og antallet af forskellige elementer i heapen N, hvor $N \leq n$.

	Ja	Nej
n	☐	☐
N	☒	☐
$n - N$	☒	☐
$n \cdot \log n$	☐	☐
$N \cdot \log n$	☐	☐
$(n - N) \cdot \log n$	☒	☐

Opgave 19 (Invarianter, 4 %)

check

Givet et ikke-negativt heltal x og et positive heltal y, så beregner nedenstående algoritme $\lfloor x/y \rfloor$.

Algoritme Division(x, y)
Inputbetingelse : Heltal $x \geq 0$ og $y \geq 1$
Outputkrav : $r = \lfloor x/y \rfloor$
Metode : $r \leftarrow 0$
$\{I\}$ **while** $x \geq y$ **do**
$x \leftarrow x - y$
$r \leftarrow r + 1$

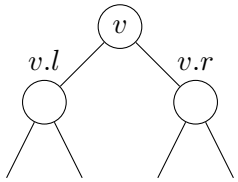
For hvert af følgende udsagn, angiv om de er en invariant I for algoritmen Division, hvor x_0 og y_0 angiver værdierne for henholdsvis x og y i starten.

	Ja	Nej
$r = \lfloor x/y \rfloor$	☐	☒
$r = \lfloor x_0/y_0 \rfloor$	☐	☒
$r = \lfloor (x_0 - x)/y \rfloor$	☒	☐
$x + ry = x_0$	☒	☐
$r(x - x_0) = y$	☐	☒

Opgave 20 (Udvidede søgetræer, 4 %)

check

For en sorteret liste af tal $x_1 \leq x_2 \leq \cdots \leq x_n$ definerer vi *sum of square gaps* (*ssg*) som $\sum_{i=2..n}(x_i - x_{i-1})^2$. Betragt et rød-sort træ hvor hver knude v gemmer et heltal $v.x$, og knuderne er ordnet venstre-mod-højre efter stigende $v.x$. Desuden gemmer v værdierne $v.min$, $v.max$ og $v.ssg$, som er hhv. det mindste, største og *sum of square gaps* af elementerne i undertræet rodet i v.

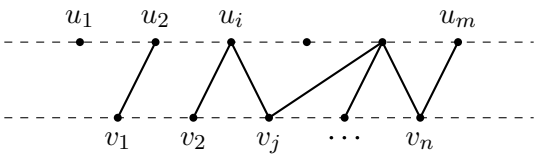


Angiv hvorledes $v.ssg$ kan beregnes når $v.min$, $v.max$ og $v.ssg$ er kendt ved de to børn $v.l$ og $v.r$ (det kan antages at disse begge eksisterer).

- $v.ssg = v.l.ssg + v.r.ssg$ ☐
- $v.ssg = v.l.ssg + (v.r.min - v.l.max)^2 + v.r.ssg$ ☐
- $v.ssg = v.l.ssg + (v.x - v.l.max)^2 + (v.x - v.r.min)^2 + v.r.ssg$ ☐
- $v.ssg = v.l.ssg + (v.r.x - v.l.x)^2 + v.r.ssg$ ☐
- $v.ssg = v.l.ssg + (v.x - v.l.x)^2 + (v.x - v.r.x)^2 + v.r.ssg$ ☐

Dynamisk programmering

De næste fire opgaver vedrører at løse *forbindelses* problemet ved hjælp af dynamisk programmering. Vi er givet to mængder af knuder $U = \{u_1, \ldots, u_m\}$ og $V = \{v_1, \ldots, v_n\}$, som ligger på to parallelle linjer fra venstre-mod-højre. Vi ønsker at forbinde par af knuder (u_i, v_j) med rette linjer, således at ingen linjer overlapper på nær i endepunkter.



Hver mulig forbindelse (u_i, v_j) har en reel værdi $w(i, j)$, muligvis negativ. Vi ønsker at finde en mængde af ikke-overlappende forbindelse med maksimal samlet værdi. For $0 \leq i \leq m$ og $0 \leq j \leq n$ lader vi $W(i, j)$ angive den maksimale værdi man kan opnå ved at forbinde $\{u_1 \ldots, u_i\}$ med $\{v_1, \ldots, v_j\}$ med ikke-overlappende forbindelse

$W(i, j)$ kan bestemmes ved følgende rekursionsformel.

$$W(i,j) = \begin{cases} 0 & i = 0 \text{ eller } j = 0 \\ \max\{0, w(i,j)\} + \max\{W(i-1,j), W(i,j-1)\} & \text{ellers} \end{cases}$$

De følgende 4 opgaver består i at udfylde 4 blokke i følgende algoritmeskabelon.

Algoritme Connect(w)
Opret tom tabel $W[0\,..\,m, 0\,..\,n]$

for ...
≪ **Opgave 21**: iterer over W ≫

≪ **Opgave 22**: beregn $W[i, j]$ ≫

≪ **Opgave 23**: sæt *solution* til maksimal værdi ≫

≪ **Opgave 24**: udskriv en løsning ≫

Opgave 21 (4 %) check

For hver af nedenstående stykker kode, angiv om det vil kunne føre til en korrekt løsning.

	Ja	Nej
for $i = 0$ **to** m **for** $j = 0$ **to** n	☒	☐
for $j = 0$ **to** n **for** $i = 0$ **to** m	☒	☐
for $i = m$ **to** 0 **step** -1 **for** $j = 0$ **to** n	☐	☒
for $j = n$ **to** 0 **step** -1 **for** $i = 0$ **to** m **step** -1	☐	☒

Opgave 22 (4 %) check

For hver af nedenstående stykker kode, angiv om det vil kunne føre til en korrekt løsning.

	Ja	Nej
$W[i,j] = \max(0, w[i,j] + \max(W[i-1,j], W[i,j-1]))$	☐	☐
$W[i,j] = 0$ **if** $i > 0$ **and** $j > 0$ **then** $W[i,j] = \max(0, w[i,j]) + \max(W[i-1,j], W[i,j-1])$	☐	☐

if $i = 0$ **or** $j = 0$ **then**
  $W[i,j] = 0$
else
  if $w[i,j] > 0$ **then**
  $W[i,j] = w[i,j]$
  else
  $W[i,j] = 0$
  $W[i,j] = W[i,j] + \max(W[i-1,j], W[i,j-1])$
	☐	☐

Opgave 23 (4 %) **check**

For hver af nedenstående stykker kode, angiv om det vil kunne føre til en korrekt løsning.

	Ja	Nej
$solution = W[0,0]$	☐	☐
$solution = W[m,n]$	☐	☐

$solution = 0$
for $i = 0$ **to** m
  for $j = 0$ **to** n
  $solution = \max\{solution, W[i,j]\}$
	☐	☐

Opgave 24 (4 %) **check**

For hver af nedenstående stykker kode, angiv om det vil kunne føre til en korrekt løsning.

	Ja	Nej

$i = m$
$j = n$
while $i > 0$ **and** $j > 0$ **do**
  if $i = j$ **then**
  print (i, j)
  if $i > j$ **then**
  $i = i - 1$
  else
  $j = j - 1$
	☐	☐

$i = m$
$j = n$
while $i > 0$ **and** $j > 0$ **do**
  if $w[i,j] > 0$ **then**
  print (i, j)
  if $W[i-1,j] > W[i,j-1]$ **then**
  $i = i - 1$
  else
  $j = j - 1$
	☐	☐

$i = 1$
$j = 1$
while $i < m$ **and** $j < n$ **do**
  if $w[i,j] > 0$ **then**
  print (i, j)
  if $W[i+1,j] > W[i,j+1]$ **then**
  $i = i + 1$
  else
  $j = j + 1$
	☐	☐