EXAM **Algorithms and Data Structures (520251E001)** **Monday, January 5, 2026, 10:00–12:00**
Institut for Datalogi, Naturvidenskabelige Fakultet, Aarhus Universitet
Number of pages (including front page): **15**
Aids: **None**

Only the separate answer sheet must be filled out and submitted.
This problem statement should *not* be submitted.

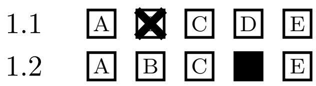
Information and Scoring

This exam set consists of a number of multiple-choice problems. The answers to the problems are indicated on the **separate answer sheet** which is to be submitted. For each problem, the share of the total exam set is indicated. Each subproblem has exactly one correct answer. For each subproblem, you may choose **at most one answer** by crossing out the corresponding box. A subproblem is graded as follows:

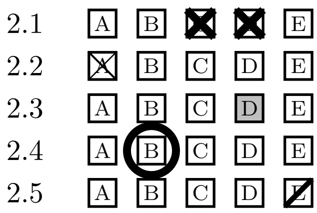
- If you mark the correct answer, you get 1 point.
- If you do not mark any answers, you get 0 points.
- If you mark an incorrect answer, you get $-\frac{1}{k-1}$ points, where k is the number of answer options.

For a problem with weight $v\,\%$ and with n subproblems, where you achieve a total of s points, your score for the problem is calculated as $s\,/\,n \cdot v\,\%$. Note that it is possible to get negative points for a problem.

Correct markings



Invalid markings



A marking must be a clear dark cross between the corners of a box, or a completely filled box. If you make a mistake, you can write next to the answer options what your answer.

Problem 1 (Asymptotic notation, 6 %)

In the following, $\log n$ denotes the base-2 logarithm of n.

	Yes	No	
$n^2 = O(2^n)$	☒	B	1.1
$3^n/2^n = O(n^2)$	A	☒	1.2
$\log(n^2) = O(2^{\log n})$	☒	B	1.3
$4n^{2/3} + 2^{3\log n} = O((\log n)^6)$	A	☒	1.4
$2n \cdot \log n = O(3^n)$	☒	B	1.5
$n \cdot \log n = O(n!)$	☒	B	1.6
$(\log n)^3 = O(\sqrt{n})$	☒	B	1.7
$n \cdot \log n + \sqrt{n}/5 = O(\sqrt{n})$	A	☒	1.8
$3n \cdot \log n = O(n \cdot \log n)$	☒	B	1.9
$\log n = \Omega(1)$	☒	B	1.10
$6 + (\log n)^3 = \Omega(n^3)$	A	☒	1.11
$2^{\log n} = \Theta(n)$	☒	B	1.12

Problem 2 (Analysis of loops, 6 %)

```
Algorithm loop1(n)
  s = 1
  for i = 1 to n
      s = s + 1
```

```
Algorithm loop2(n)
  s = 1
  for i = 1 to n
      for j = 1 to i
          s = s + 1
```

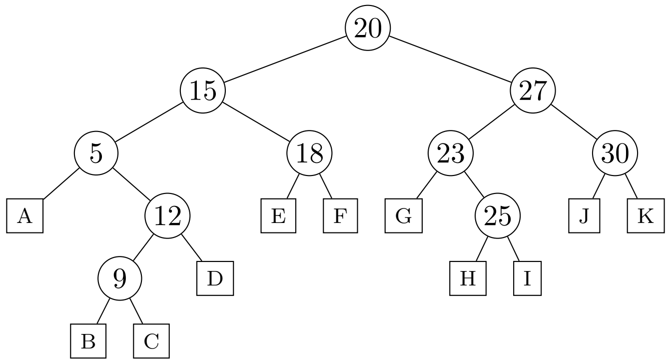
```
Algorithm loop3(n)
  i = 0
  s = 0
  while s ≤ n * n
      i = i + 1
      s = s + i
```

```
Algorithm loop4(n)
  i = 2
  while i ≤ n
      j = i
      while j ≤ n
          j = j + 1
      i = i * i
```

Indicate for each of the above algorithms the running time as a function of n in Θ-notation.

	$\Theta(n^2)$	$\Theta(\log n)$	$\Theta(n)$	$\Theta(n^3)$	$\Theta(n \cdot \log\log n)$	$\Theta((\log n)^2)$	$\Theta(n \log n)$	$\Theta(\sqrt{n})$	
loop1	A	B	☒	D	E	F	G	H	2.1
loop2	☒	B	C	D	E	F	G	H	2.2
loop3	A	B	☒	D	E	F	G	H	2.3
loop4	A	B	C	D	☒	F	G	H	2.4

Problem 3 (Insertions into search trees, 4 %)



Indicate in which leaves A–K in the above unbalanced binary search tree the elements 4, 29, 31, 22 and 14 should be inserted (it is assumed that before each insertion the tree contains only the above ten elements).

	A	B	C	D	E	F	G	H	I	J	K	
INSERT(4)	☒	B	C	D	E	F	G	H	I	J	K	3.1
INSERT(29)	A	B	C	D	E	F	G	H	I	☒	K	3.2
INSERT(31)	A	B	C	D	E	F	G	H	I	J	☒	3.3
INSERT(22)	A	B	C	D	E	F	☒	H	I	J	K	3.4
INSERT(14)	A	B	C	☒	E	F	G	H	I	J	K	3.5

Problem 4 (Max-Heap-Insert, 4 %)

What is the binary max-heap after inserting the elements 8, 6, 2, 11, 7, 10 and 14 in the given order with MAX-HEAP-INSERT, starting with the empty heap.

1	2	3	4	5	6	7	
14	11	10	8	7	6	2	A

1	2	3	4	5	6	7	
14	8	11	6	7	2	10	☒

1	2	3	4	5	6	7	
8	11	14	6	7	10	2	C

1	2	3	4	5	6	7	
14	11	10	6	7	8	2	D

1	2	3	4	5	6	7	
8	6	2	11	7	10	14	E

Problem 5 (Heap-Extract-Max, 4 %)

1	2	3	4	5	6	7	8	9	10	11	12	13
26	24	20	17	21	19	1	5	14	8	4	7	11

What is the resulting max-heap after applying HEAP-EXTRACT-MAX to the above max-heap?

1	2	3	4	5	6	7	8	9	10	11	12	
24	21	17	20	19	11	5	14	8	4	7	1	A

1	2	3	4	5	6	7	8	9	10	11	12	
24	21	20	17	8	19	1	5	14	11	4	7	B

1	2	3	4	5	6	7	8	9	10	11	12	
24	21	20	17	11	19	1	5	14	8	4	7	☒

1	2	3	4	5	6	7	8	9	10	11	12	13	
24	21	20	17	8	19	1	5	14		4	7	11	D

1	2	3	4	5	6	7	8	9	10	11	12	
24	21	20	17	8	19	1	5	14	4	7	11	E

Problem 6 (Merge-Sort, 4 %)

Algorithm MERGE-SORT(A, p, r)
  if $r - p + 1 \leq k$
    INSERTION-SORT(A, p, r)
  else
    $q = \lfloor (p + r)/2 \rfloor$
    MERGE-SORT(A, p, q)
    MERGE-SORT$(A, q + 1, r)$
    MERGE(A, p, q, r)

Consider the above variant of the MERGE-SORT algorithm, which uses INSERTION-SORT to sort subarrays of size at most k. What is the running time of this algorithm in Θ-notation as a function of both n and k, where n is the size of the array to be sorted?

$\Theta(n \log(nk))$	$\Theta(nk \log n)$	$\Theta(nk + n \log n)$	$\Theta(k^2 + n \log n)$	$\Theta(nk + n((\log n) - k))$
A	B	☒	D	E

Problem 7 (Linear probing, 4 %)

0	1	2	3	4	5	6	7	8	9	10
			5			21	10		15	13

In the above hash table of size 11, *linear probing* is used with hash function $h(k) = 5k \bmod 11$. Indicate the positions the five elements 2, 7, 8, 9 and 11 will be inserted at in the hash table (for each insertion, we assume the hash table only contains the elements 5, 10, 13, 15 and 21).

	0	1	2	3	4	5	6	7	8	9	10	
INSERT(2)	☒ A	B	C	D	E	F	G	H	I	J	K	7.1
INSERT(7)	A	B	☒ C	D	E	F	G	H	I	J	K	7.2
INSERT(8)	A	B	C	D	E	F	G	H	☒ I	J	K	7.3
INSERT(9)	A	☒ B	C	D	E	F	G	H	I	J	K	7.4
INSERT(11)	☒ A	B	C	D	E	F	G	H	I	J	K	7.5

Problem 8 (Quadratic probing, 4 %)

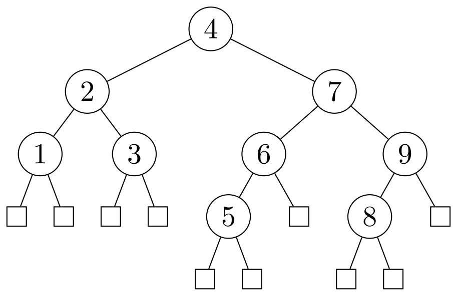
0	1	2	3	4	5	6	7	8	9	10
11			22					6	4	13

In the above hash table of size 11, *quadratic probing* is used with hash functions $h'(k) = 5k \bmod 11$ and $h(k, i) = (h'(k) + 2i + i^2) \bmod 11$. Indicate the positions the five elements 1, 2, 5, 8 and 9 will be inserted at in the hash table (for each insertion, we assume the hash table only contains the elements 4, 6, 11, 13 and 22).

	0	1	2	3	4	5	6	7	8	9	10	
INSERT(1)	A	B	C	D	E	☒ F	G	H	I	J	K	8.1
INSERT(2)	A	B	☒ C	D	E	F	G	H	I	J	K	8.2
INSERT(5)	A	B	C	D	E	F	☒ G	H	I	J	K	8.3
INSERT(8)	A	B	C	D	E	F	G	☒ H	I	J	K	8.4
INSERT(9)	A	☒ B	C	D	E	F	G	H	I	J	K	8.5

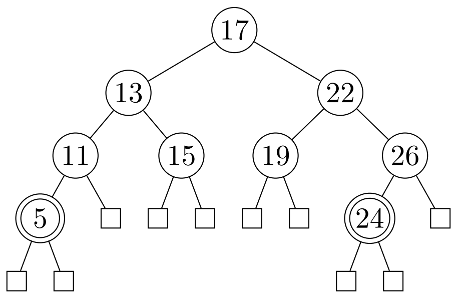
Problem 9 (Valid red-black trees, 4 %)

For each of the following subsets, indicate whether the binary tree below is a valid red-black tree if these nodes are colored red and all other nodes black.

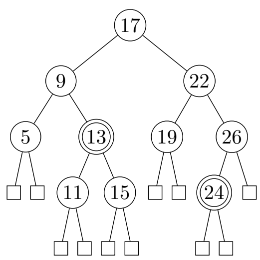


	Yes	No	
1, 3, 5, 6, 8, 9	A	☒	9.1
4, 5, 8	A	☒	9.2
1, 3, 5, 7, 8	☒	B	9.3
2, 5, 7, 8	☒	B	9.4
5, 8	☒	B	9.5

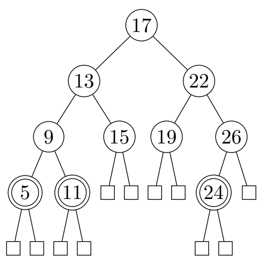
Problem 10 (Red-black tree insertion, 4 %)



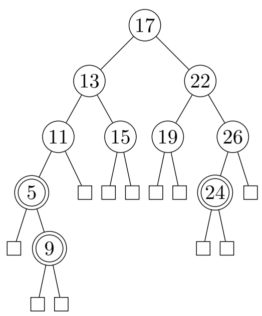
Indicate the resulting red-black tree when inserting 9 into the above red-black tree (double circles indicate red nodes).



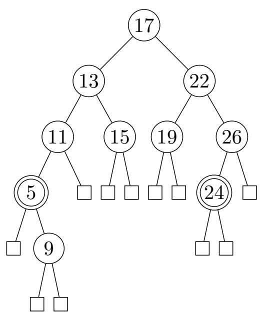
A



☒



C

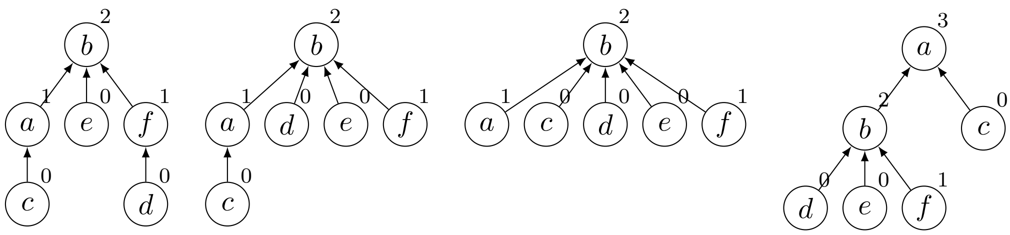


D

Problem 11 (Union-find, 4 %)

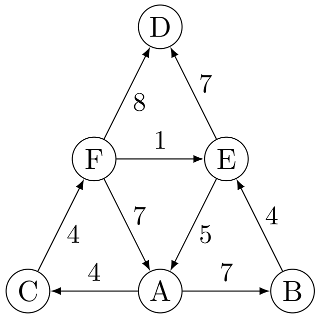
Indicate the resulting union-find structure after the following sequence of operations, when using union-by-rank and path compression.

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



☐ A ☒ B ☐ C ☐ D

Problem 12 (Dijkstra's algorithm, 4 %)

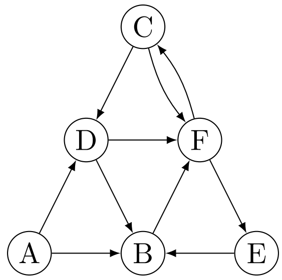


Assume Dijkstra's algorithm is used to find shortest distances from **node A** to all nodes in the graph above. Indicate the order in which the nodes are removed from the priority queue in Dijkstra's algorithm.

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

☐ A ☐ B ☐ C ☒ D

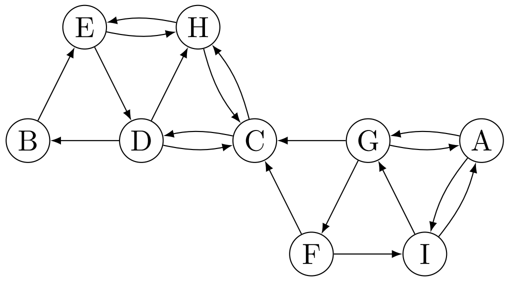
Problem 13 (BFS, 4 %)



For a breadth first search (BFS) on the graph above **starting in node A**, indicate the order in which the nodes are removed from the queue Q in the BFS algorithm. It is assumed that the graph is given by alphabetically sorted adjacency lists.

A B D F E C	A B D F C E	A D B F C E	A B F C D E
A	☒	C	D

Problem 14 (DFS, 4 %)



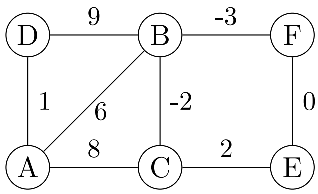
Consider a depth first search (DFS) on the graph above starting in **node A**, and where the outgoing edges of a node are visited in alphabetical order. Indicate the order in which the nodes receive their **finishing time**.

B D E H C F G I A	E B H D F C I G A	H E B D C I F G A	I F H E B D C G A	
A	B	☒	D	14.1

Indicate for each of the edges below which type the edge is in the DFS search.

	Tree edge	Back edge	Cross edge	Forward edge	
(F, C)	A	B	☒	D	14.2
(I, A)	A	☒	C	D	14.3
(G, C)	☒	B	C	D	14.4
(A, I)	A	B	C	☒	14.5

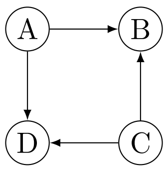
Problem 15 (Kruskal's algorithm, 4 %)



Assume Kruskal's algorithm is used to find a minimum spanning tree for the graph above. Indicate which edge is **last** included in the minimum spanning tree.

(E , F)	(C , E)	(A , B)	(B , C)	(A , D)
A	B	☒	D	E

Problem 16 (Topological sorting, 4 %)



For each of the node orderings below, state whether it is a valid topological sort of the above graph.

	Yes	No	
C A B D	☒	B	16.1
A D B C	A	☒	16.2
B A D C	A	☒	16.3
A C D B	☒	B	16.4
A C B D	☒	B	16.5

Problem 17 (Recurrence relations, 4 %)

State the solution for each of the below recurrence relations, where $T(n) = 1$ for $n \leq 1$.

	$\Theta(\log n)$	$\Theta(\sqrt{n})$	$\Theta(n)$	$\Theta(n \log n)$	$\Theta(n^2)$	$\Theta(n^2 \log n)$	$\Theta(n^3)$	
$T(n) = 9 \cdot T(n/3) + 3$	A	B	C	D	☒	F	G	17.1
$T(n) = T(n-1) + n^2$	A	B	C	D	E	F	☒	17.2
$T(n) = 8 \cdot T(n/2) + n$	A	B	C	D	E	F	☒	17.3
$T(n) = T(n/3) + 5$	☒	B	C	D	E	F	G	17.4
$T(n) = T(n-2) + 1$	A	B	☒	D	E	F	G	17.5

Problem 18 (Sorting, 4 %)

Given an array $A[1..n]$, the below algorithm sorts A when calling SortIt$(A, 1, n)$. Indicate the algorithm's running time as a function of n.

Algorithm SortIt(A, i, j)
  if $j = i + 1$ **and** $A[i] > A[j]$
    swap $A[i]$, $A[j]$
  else if $j \geq i + 2$
    $s = \lfloor (j - i + 1)/3 \rfloor$
    SortIt$(A, i, j - s)$
    SortIt$(A, i + s, j)$
    SortIt$(A, i, j - s)$

$\Theta(n)$	$\Theta(n \log n)$	$\Theta(n^{3/2})$	$\Theta(n^{\log_2 3})$	$\Theta(n^{\log_{3/2} 3})$	$\Theta(n^2)$
A	B	C	D	☒	F

Problem 19 (Invariants, 4 %)

The below algorithm PairSum(A, x) determines if a value x can be obtained as the sum of two entries $A[i] + A[j]$ from a sorted array $A[1..n]$ (possibly the same entry twice).

Algorithm PairSum(A, x)
Input : x, $A[1..n]$: $A[1] \leq \cdots \leq A[n]$
Output : i, j: $(1 \leq i \leq j \leq n \;\wedge\; A[i] + A[j] = x) \vee$
$(i > j \;\wedge\; (\forall i', j' : 1 \leq i' \leq j' \leq n \Rightarrow A[i'] + A[j'] \neq x))$
Method : $i = 1$
  $j = n$
  $\{I\}$ **while** $i \leq j$ **and** $A[i] + A[j] \neq x$
    if $x < A[i] + A[j]$ **then**
      $j = j - 1$
    else
      $i = i + 1$

For each of the following statements, indicate whether it is an invariant I for the algorithm PairSum.

Statement	Yes	No	
$1 \leq i \leq n$	A	☒	19.1
$1 \leq i \leq j + 1 \leq n + 1$	☒	B	19.2
$A[i] + A[j] = x$	A	☒	19.3
$\forall i', j' : 1 \leq i' \leq i \wedge j \leq j' \leq n \Rightarrow A[i'] + A[j'] \neq x$	A	☒	19.4
$\forall i', j' : (1 \leq i' < i \wedge 1 \leq j' \leq n) \vee (1 \leq i' \leq n \wedge j < j' \leq n) \Rightarrow A[i'] + A[j'] \neq x$	☒	B	19.5

Problem 20 (Huffman codes, 4 %)

Letter	a	b	c	d	e	f	g
Frequency	20	40	10	10	60	50	50

Assume we have a file with the above letters and frequencies. If one constructs a Huffman tree for these frequencies, what are the lengths of the codes for `a`, `c`, and `e`?

	1	2	3	4	5	6	
`a`	A	B	C	☒	E	F	20.1
`c`	A	B	C	D	☒	F	20.2
`e`	A	☒	C	D	E	F	20.3

The file contains 20 + 40 + 10 + 10 + 60 + 50 + 50 = 240 letters. How many bits would a Huffman encoding of the string use?

480	560	600	620	670	710	
A	B	C	☒	E	F	20.4

Problem 21 (Amortized analysis, 4 %)

Algorithm PUSH(Q, x)
  INSERT(I, x)

Algorithm REMOVE(Q, x)
  INSERT(D, x)
  if $2 * |D| > |I|$
    rebuild $I = I - D,\ D = \emptyset$

Algorithm POP(Q)
  while $|D| > 0 \land$ MIN(I)=MIN(D)
    EXTRACT-MIN(D)
    EXTRACT-MIN(I)
  $v =$ EXTRACT-MIN(I)
  if $2 * |D| > |I|$
    rebuild $I = I - D,\ D = \emptyset$
  return v

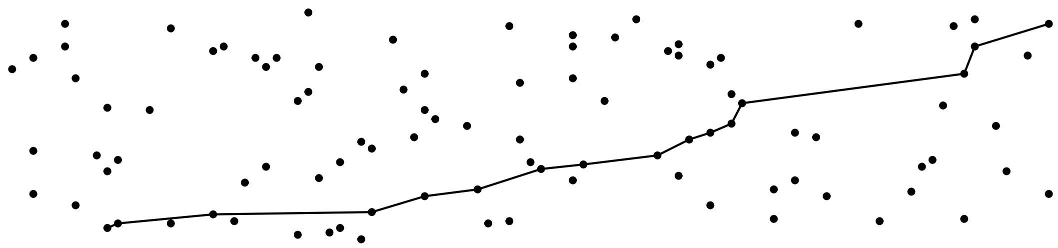
We wish to design a priority queue, which supports the three operations PUSH(Q, x), POP(Q) and REMOVE(Q, x), which respectively inserts an element x into Q, returns and removes the smallest element in Q, and removes a given element x from Q (which must be found in Q).

We represent Q by two binary heaps I and D, and implement the operations as above, where the *rebuild* operation builds a new binary heap I containing all elements in I which are not in D using sorting in time $O(|I| \log |I|)$. Recall that a binary heap H supports the operations INSERT(H, x) and EXTRACT-MIN(H) in time $O(\log |H|)$, and MIN(H) in time $O(1)$.

For each of the following functions, indicate whether it is a potential function with which one can argue that each of the three operations PUSH, POP and REMOVE take amortized $O(\log n)$ time, where n is the number of elements in Q.

	Yes	No	
$\lvert I\rvert \log \lvert I\rvert$	☒	B	21.1
$\lvert D\rvert \log \lvert D\rvert$	A	☒	21.2
$\lvert D\rvert \log \lvert I\rvert$	☒	B	21.3
$(\lvert I\rvert + \lvert D\rvert) \cdot \log(\lvert I\rvert + \lvert D\rvert)$	☒	B	21.4
$(\lvert I\rvert - \lvert D\rvert) \cdot \log(\lvert I\rvert - \lvert D\rvert)$	A	☒	21.5

Dynamic programming



Given n points $(x_1, y_1), \ldots, (x_n, y_n)$, where $x_1 \leq \cdots \leq x_n$, we want to find a subsequence of the points that is strictly increasing with respect to both the x- and y-coordinates, and contains the largest number of points. If $L(i)$ denotes maximum number of points in a strictly increasing subsequence starting at (x_i, y_i), then $L(i)$ can be determined using the following recurrence relation.

$$L(i) = \max\Big(\{1\} \cup \{1 + L(j) \mid i < j \leq n \ \wedge \ x_i < x_j \ \wedge \ y_i < y_j\}\Big)$$

The following 3 problems consist of filling in 3 boxes in the following algorithm template, where L stores the above $L(i)$ values and J is used to construct an optimal solution.

Algorithm IncreasingPoints($(x_1, y_1), \ldots, (x_n, y_n)$)
Create table $L[1\,..\,n]$, where all entries have the value 1
Create table $J[1\,..\,n]$, where all entries have the value 0

for . . .
$\ll$ **Problem 22**: iterate over L $\gg$

$\ll$ **Problem 23**: compute $L[i]$ $\gg$

$\ll$ **Problem 24**: print longest increasing subsequence of points $\gg$

Problem 22 (4 %)

For each of the code fragments below, indicate whether it could lead to a correct solution.

	Yes	No	
for $i = 1$ **to** n	A	☒	22.1
for $i = n$ **to** 1 **step** -1	☒	B	22.2

Problem 23 (4 %)

For each of the code fragments below, indicate whether it could lead to a correct solution.

	Yes	No	

for $j = i + 1$ **to** n
  if $x_i < x_j \ \wedge\ y_i < y_j \ \wedge\ L[i] < 1 + L[j]$ **then**
    $L[i] = 1 + L[j]$
    $J[i] = j$
	☒	B	23.1

for $j = i + 1$ **to** n
  if $x_i < x_j \ \wedge\ y_i < y_j$ **then**
    $L[i] = \max(L[i], 1 + L[j])$
    $J[i] = j$
	A	☒	23.2

for $j = n$ **to** $i + 1$ **step** -1
  if $x_i < x_j \ \wedge\ y_i < y_j \ \wedge\ L[i] \leq L[j]$ **then**
    $L[i] = 1 + L[j]$
    $J[i] = j$
	☒	B	23.3

Problem 24 (4 %)

For each of the code fragments below, indicate whether it could lead to a correct solution.

	Yes	No	

$i = 1$
while $i > 0$
  print (x_i, y_i)
  $i = J[i]$
	A	☒	24.1

$i = 1$
for $j = 2$ **to** n
  if $L[j] > L[i]$ **then**
    $i = j$
while $i > 0$
  print (x_i, y_i)
  $i = J[i]$
	☒	B	24.2

$i = J[1]$
while $i > 0$
  print (x_i, y_i)
  $i = J[i]$
	A	☒	24.3