UNIVERSITY OF LONDON

CM2035

BSc EXAMINATION

COMPUTER SCIENCE

Algorithms and Data Structures II

Release date: Tuesday 13 September 2022 at 12:00 midday British Summer Time

Submission date: Wednesday 14 September 2022 by 12:00 midday British Summer Time

Time allowed: 24 hours to submit

INSTRUCTIONS TO CANDIDATES:

Section A of this assessment paper consists of a set of **TEN** Multiple Choice Questions (MCQs) which you will take separately from this paper. You should attempt to answer **ALL** the questions in Section A. The maximum mark for Section A is **40**.

Section A will be completed online on the VLE. You may choose to access the MCQs at any time following the release of the paper, but once you have accessed the MCQs you must submit your answers before the deadline or within **4 hours** of starting, whichever occurs first.

Section B of this assessment paper is an online assessment to be completed within the same 24-hour window as Section A. We anticipate that approximately **1 hour** is sufficient for you to answer Section B. Candidates must answer **TWO** out of the THREE questions in Section B. The maximum mark for Section B is **60**.

Calculators are not permitted in this examination. Credit will only be given if all workings are shown.

You should complete **Section B** of this paper and submit your answers as **one document**, if possible, in Microsoft Word or a PDF to the appropriate area on the VLE. Each file uploaded must be accompanied by a coversheet containing your **candidate number** written clearly at the top of the page before you upload your work. Do not write your name anywhere in your answers.

SECTION A

Candidates should answer the **TEN** Multiple Choice Questions (MCQs) quiz, **Question 1** in Section A on the VLE.

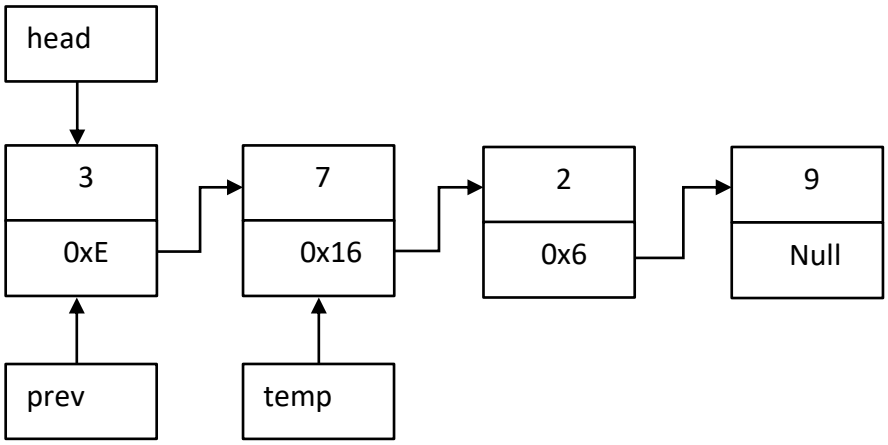
SECTION B

Candidates should answer any **TWO** questions in Section B.

Question 2

This question concerns linked links.

(a) Consider the following linked list and pointers `prev` and `temp`.



Redraw the list after execution of the pseudocode: `prev -> next = temp -> next`.

[4]

(b) Explain how the pseudocode in part (a) can be used to delete a node from a linked list.

[4]

(c) The function `delete(list, x)` removes the node with value x from the linked list `list`. A protype for `delete` is:

```
function delete(list, x)
     Node temp = head
     Node prev
     // 1. code to print a warning if the list is empty
     // 2. code to remove the first node from the list
     // 3. code to remove an arbitrary (not first) node
     // 4. code to print a warning if x is not found
end function
```

Each block 1-4 returns the list, either modified or unchanged.

(i) Provide an implementation of code block 1.

[4]

(ii) Provide an implementation of code block 2.

[4]

(iii) An (incorrect) implementation of Code block 3 is

```
while (temp != Null)
     if (temp -> data == x)
          prev -> next = temp -> next
          return list
```

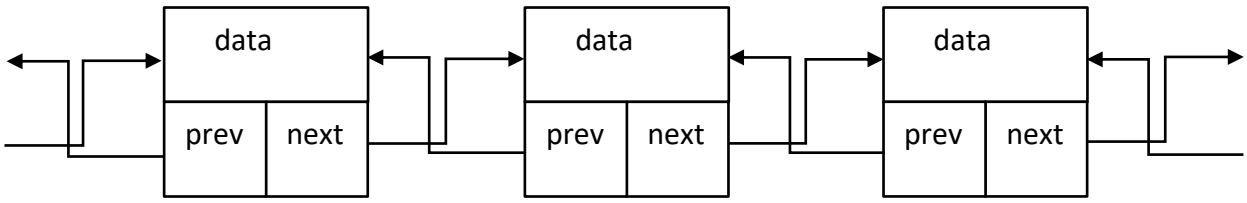
but the code does not perform as intended. Explain why the code does not work and provide a correction.

[4]

(d) Write a pseudocode function `SEARCH(L, k)` that takes a pointer `L` to the head of a linked list and a value `k` as parameters and returns a pointer to the node with value `k`. The function should return `Null` if the value is not found. (Assume that all values in the list are distinct.)

[4]

(e) Nodes in a *doubly linked* list contain pointers `prev` and `next` to nodes either side:



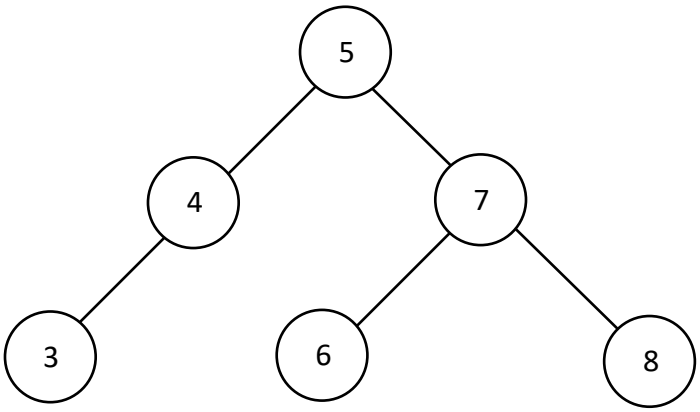
Suppose that `x` is a pointer to a node in a doubly linked list. Write a function `DELETE(L, x)` that removes node `x` from a doubly linked list headed by `L`.

[6]

Question 3

This question concerns binary search trees (BSTs).

(a) Consider the binary search tree in the diagram below.



Suppose the tree is traversed is in-order and the content of each node is printed when visited. What is printed to the screen?

[4]

(b) Write pseudocode for a function `IN-ORDER` that takes the root of a BST as a single argument and prints the tree in pre-order.

[4]

(c) Write a pseudocode function `TREE-MIN` that returns the minimum-value node of a BST. (The minimum-value node of the BST in part (a) is the node with value 3.)

[4]

(d) BST implementations frequently enhance each node with an additional pointer to the node's parent i.e. `x -> parent` points to `x`'s parent. The parent of the root is `Null`.

Suppose x is an arbitrary BST node. What is returned by the function R(x) defined below?

[4]

```
function R(x)
     y = x -> parent
     while(y != Null)
          x = y
          y = x -> parent
     end while
     return y
end function
```

(e) What is returned by the following function? [4]

```
function RR(x)
     y = x -> parent
     while(y != Null && x == y -> right)
          x = y
          y = x -> parent
     end while
     return y
end function
```

(f) Consider the following function:

```
function S(x)
     if x -> right != Null
          return TREE-MIN(x -> right)
     y = x -> parent
     while (y != Null and x == y -> right)
          x = y
          y = y -> parent
     return y
end function
```

and the BST in part (a).

What is `S(x)` when `x` is the node with:

(i) value 3?
(ii) value 8?

[4]

(g) It is claimed that `S(x)` in part (f) returns the successor (smallest node that is larger) to `x` or `null` if `x` is the largest node in the BST? Is this claim correct. Explain your reasoning.

[6]

Question 4

This question concerns topics from part 1 of the syllabus.

(a) Explain an advantage of Theta-notation over big-O notation. Illustrate your answer with an example.

[6]

(b) What is the running time, T(n) of EXP? Show your working.

[4]

```
function EXP(n)
     if (n == 0) return 1;
     else return n * EXP(n – 1)
end function
```

(c) Explain a situation where it is better to use Counting Sort rather than Merge Sort to sort an array in ascending order.

[4]

(d) Explain a situation where it is better to use Merge Sort rather than Counting Sort to sort an array in ascending order.

[4]

(f) Write an account of how hash table collisions can be resolved.

[12]

END OF PAPER