CM2035

BSc EXAMINATION

COMPUTER SCIENCE

Algorithms and Data Structures II

Release date: Monday 14 September 2020: 12.00 midday British Summer Time

Time allowed: 24 hours to submit

Submission date: Tuesday 15 September 2020: 12.00 midday British Summer Time

INSTRUCTIONS TO CANDIDATES:

Part A of this assessment consists of a set of 10 Multiple Choice Questions (MCQs) which you will take separately from this paper. You should attempt to answer **ALL** the questions in Part A. The maximum mark for Part A is **40**.

Part 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. Candidates only have **ONE** attempt at Part A.

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

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

You should complete **Part 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. You are permitted to upload 30 documents. However, we advise you to upload as few documents as possible. Each file uploaded must be accompanied by a coversheet containing your candidate number. In addition, your answers must have your **candidate number** written clearly at the top before you upload your work. Do not write your name anywhere in your answers.

PART B

Candidates should answer any **TWO** questions from Part B.

Question 1

The following algorithms, A1 and A2, solve the same problem. To do so, they receive as input arguments:

root: the root of a binary search tree (BST) storing integer numbers
x: an integer number

ALGORITHM 1

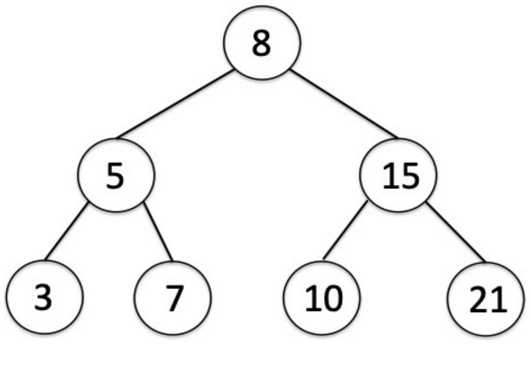
```
function A1(root, x)
   Q = new Queue()
   ENQUEUE(Q,root)
   while !ISEMPTY(Q)  do
        t = PEEK(Q)
        if (t==x)
             return TRUE
        else
        ENQUEUE(Q,left(t))
        ENQUEUE(Q,right(t))
        DEQUEUE(Q)
   end while
   return FALSE
end function
```

Note: the function ENQUEUE only inserts a new element in the queue if this element is different from NULL

ALGORITHM 2

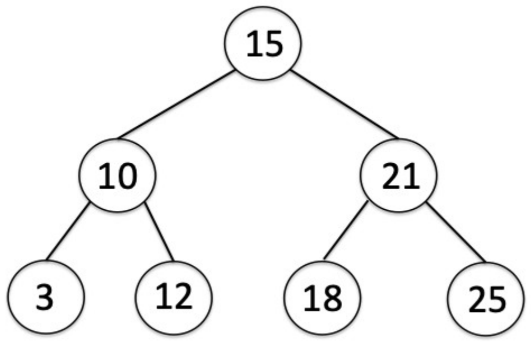
```
function A2(root,x)
   if (root==NULL)
      return FALSE
   else
      if(x == root->data)
          return TRUE
      else
          if (x< root->data)
                 return  A2(root->left,x)
          else
                 return  A2(root->right,x)
end function
```

(a) For the following BST:



What is the content of the queue **immediately before returning** from the execution of A1(root,21)? [4]

(b) For the following BST: [4]



What is the return value of A2(root, 20)?

(c) What is the task performed by algorithms A1 and A2? Don´t forget to mention the return values for the different cases [4]

(d) What is the worst-case time complexity of A1? Use Theta notation [1] and explain your reasoning [3] [4]

(e) Assuming a fully balanced BST (a BST with all its levels fully populated) of N elements, what is the recurrence equation describing the running time of A2? [4]

(f) What is the worst-case time complexity of A2? Use Theta notation [1] and show your workings [3] [4]

(g) Which algorithm do you recommend implementing? Why? [6]

Question 2

The data structure shown in Figure 1 can be thought as a list of two lists. In this case, the data structure is used to classify numbers in one of 2 groups: even numbers and odd numbers.

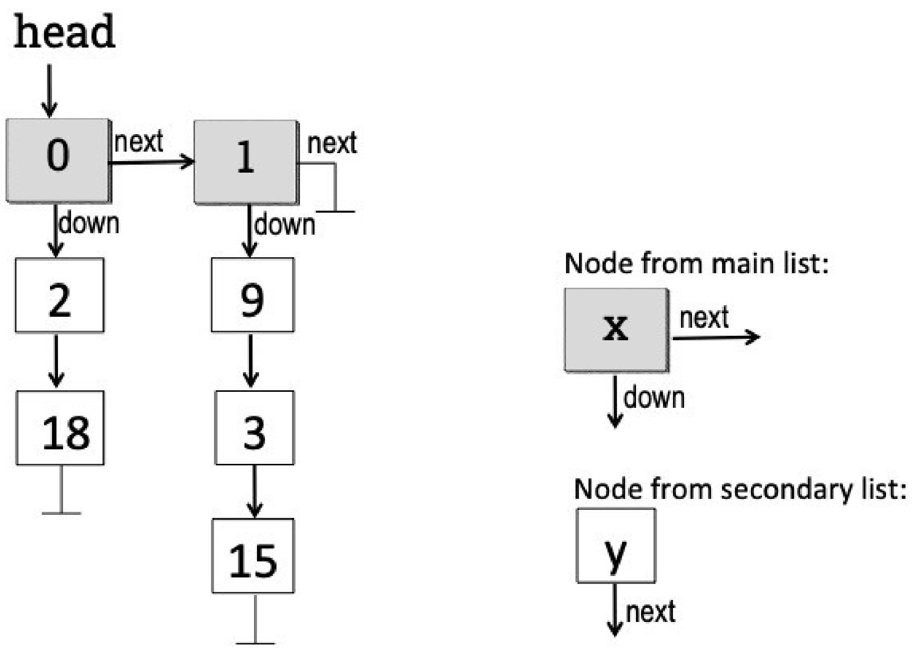


Figure 1. A list of two lists

The first node of the main list (the list drawn horizontally, made of shaded nodes) stores a number 0 to signal that the numbers stored in its secondary list (the list drawn vertically, 'hanging' from node storing number 0) are even (in the figure, the numbers 2 and 18). The second node of the main list stores a number 1, to signal that the numbers stored in its secondary list (in the figure, the numbers 9, 3 and 15) are odd.

(a) In a single linked list implemented in C++ this is the definition of a node: [6]

```
struct Node {
      int data;
      Node *next;
  };
```

This definition is useful for the nodes belonging to the secondary lists, but not for the nodes belonging to the main list. Main list nodes need two pointers (one for the next node in the main list and one for the first node in the secondary list). Assuming that the above definition is kept for the nodes of the secondary lists, propose a new definition of node for the nodes of the main list. Call this type of node Node_main and use the names of pointers given in Figure 1.

(b) Write the pseudocode of the function INSERT(head,x) that inserts a new number in this data structure. If the number is even it must go to the first secondary list (the one ‘hanging’ from node 0 in the main list). Otherwise, it must go to the second secondary list. Assume the data structure already has the main list created and numbers are inserted at the start of the secondary list. [8]

(c) Write the pseudocode of the function SEARCH(head,x) that returns TRUE if the number x is in the data structure (in any of the secondary lists) and FALSE otherwise. [8]

(d) Write the pseudocode of the function DELETE(head, b) that receives as input arguments the head of the last of two lists and a Boolean value. The function DELETE(head,b) deletes one of the main nodes. If b equals 0, then the node storing number 0 is deleted. Otherwise, the main node storing number 1 is deleted. Consider the following cases: the main list is empty and it has only one node (node 0 or 1). [8]

Question 3

A software developer needs to solve the following problem: given the adjacency matrix of an **undirected weighted** graph, find the value of the k-th minimum cost edge. Assume that **all edge weights are different**, that they are non-negative integer numbers, and that they are not greater than 999. The number 1000 (one thousand) signals the absence of an edge.

For example, for the graph represented by the following adjacency matrix M:

	A	**B**	**C**	**D**
A	1000	3	1	5
B	3	1000	7	4
C	1	7	1000	9
D	5	4	9	1000

The first minimum (that is, k=1) is 1, the second minimum (k=2) is 3, the third minimum (k=3) is 4, and so on.

The design of the algorithm must take as input arguments the adjacency matrix (M), its number of nodes (N) and the value of k. It must return the value of the k-th minimum cost edge.

The software developer came up with these two algorithms to solve the problem:

Algorithm 1:

1. Make a copy of the adjacency matrix. Call the copy M_copy.
2. Create a variable, called `min`, where the minimum value is recorded
3. Create a variable, called `count`. Initialise its value to 0
4. Visit every element of M_copy, from top to bottom, from left to right and record the minimum value in `min`
5. Once the minimum value is found, increase the variable `count` by one unit
6. If the condition `count==k` is true, return the value of `min`. Otherwise, delete the minimum value from M_copy (write number 1000 in the corresponding positions) and repeat steps 2-6

Algorithm 2:

- Create a min-heap storing the values of all edges
- Perform EXTRACT_MIN k times. The value last extracted is the k-th minimum.

(a) Write the pseudocode of Algorithm 1. [8]

(b) Write the pseudocode of Algorithm 2. Assume you already have implemented the following min-heap functions:

- INSERT(heap,x):insert number x into the heap. Worst-case Theta(logN)
- BUILD_HEAP(A): build a min heap in place. Worst-case Theta(N)
- EXTRACT_MIN(heap): return the minimum value stored in the min-heap. Worst-case Theta(logN)

That is, you can use these functions with no need to write the pseudocode for them. [8]

(c) What are the worst-case time complexities of A1 and A2? Use Theta-notation. Justify your findings. [8]

(d) What algorithm do you recommend for implementation? Justify in terms of worst-case time complexity. [6]

END OF PAPER