UNIVERSITY OF LONDON

CM1035

BSc EXAMINATION

COMPUTER SCIENCE

Algorithms and Data Structures I

Release date: Monday 21 September 2020: 12.00 midday British Summer Time

Time allowed: 24 hours to submit

Submission date: Tuesday 22 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.

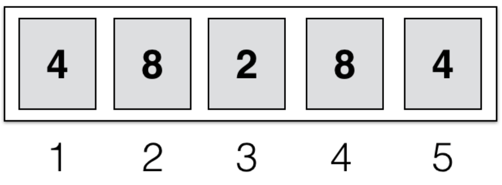
UL20/0953

PART B

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

Question 1 This question is about checking if a vector is a palindrome. A vector is a palindrome if it is exactly the same after the order of the elements is reversed (the vector is flipped). An empty vector is a palindrome.

(a) The following vector is an example of a palindrome:



Give an example of a vector that is **not** a palindrome. [3]

(b) To check if a vector of length n is a palindrome, one can first create an empty stack, and then push the first floor(n/2) values in the vector to it. Here floor(x) means the largest integer smaller than or equal to x. Consider the following piece of pseudocode:

```
function MAKESTACK(vector)
    new Stack s
    n ← LENGTH(vector)
    for $1 \leq i \leq$ floor(n/2) do
        PUSH[vector[i], s]
    end for
    return s
end function
```

Draw a picture of the returned stack after calling this function MAKESTACK on the example palindrome vector provided in part (a) of this question. [4]

(c) A linked list can be used to implement the stack in part (b) of this question. Draw the corresponding linked list of the final stack in part (b). [4]

(d) To complete the algorithm for deciding if a vector is a palindrome, after creating the stack using MAKESTACK, the contents of the stack are compared with the remaining values in the vector, starting from index ceil(n/2) + 1. Here ceil(x) is the smallest integer larger than or equal to x (the ceiling). Consider the following piece of incomplete pseudocode:

```
function IsPalindrome(vector)
    if LENGTH(vector) = 1 then
        return TRUE
    end if
    new Stack $s \leftarrow$ MakeStack(vector)
    n $\leftarrow$ LENGTH(vector)
    for ceil(n/2) + 1 $\leq$ i $\leq$ n do
        if TOP[$s$] $\neq$ vector[i] then
            return MISSING1
        end if
        MISSING2
    end for
    return TRUE
end function
```

This function should return TRUE if the input vector is a palindrome, or return FALSE otherwise. What should go in the place of MISSING1 and MISSING2 to complete this function? [4]

(e) Provide an argument for why the worst-case time complexity of implementing IsPalindrome is O(n) for an input vector of length n. [5]

(f) Another method for deciding if an input vector is a palindrome is through recursion. Consider the following piece of incomplete pseudocode:

```
function IsPalNew(()vector)
    n $\leftarrow$ LENGTH(vector)
    return RecPalindrome(vector, 1, n)
end function
function RecPalindrome(vector, left, right)
    if right $\leq$ left then
        return TRUE
    end if
    if vector[left] $\neq$ vector[right] then
        return MISSING1
    end if
    MISSING2
end function
```

The function IsPalNew should return TRUE if the input vector is a palindrome, and FALSE otherwise - the function calls RecPalindrome to do this.

i. What should go in the place MISSING1? [2]

ii. What should go in the place of MISSING2? [4]

(g) Briefly explain why the worst-case time complexity of implementing ISPALNEW is also O(n) for an input vector of length n. [4]

Question 2 This question is about sorting algorithms.

(a) Consider the following vector of integers:

4	**2**	**12**	**1**	**7**	**9**	**9**
1	2	3	4	5	6	7

i. By hand, work through the standard bubble sort algorithm on this vector, explicitly showing how it changes in the algorithm. You should sort the vector in ascending order so the lowest value is in the first element. [7]

ii. What is the worst-case time complexity in n of the Bubble Sort algorithm for a vector of length n? [3]

(b) Consider the following vector of integers:

6	**2**	**3**	**5**	**4**
1	2	3	4	5

i. By hand, work through the Quicksort algorithm on this vector where the pivot is the mid-point (calculated as the floor of half the sum of the left index plus the right index). Explicitly show how the vector changes in the algorithm. You should sort the vector in ascending order so the lowest value is in the first element. [6]

ii. Give an example of a worst-case input vector of four elements for Quicksort where the pivot is the mid-point. Each element should store an integer and assume sorting in ascending order. Very briefly explain why it is a worst-case input. [5]

iii. Briefly explain the role of recursion in the Quicksort algorithm. [4]

(c) Consider the following piece of incomplete JavaScript:

```
function swap(array, i, j) {
  var store = array[i];
  array[i] = array[j];
  array[j] = store;
  return array;
}

function sort(array, right) {
  if (right <= 0) {
    return MISSING1;
  }
  for (var i = 0; i < right; i++) {
    if (array[i+1] < array[i]){
      swap(array, i, i+1);
    }
  }
  return MISSING2;
}

function sorter(array) {
  return sort(array, array.length - 1);
}
```

When completed this code should implement a recursive version of a sorting algorithm on a JavaScript array. The function `sorter` will return the sorted version of the argument `array`.

i. What should go in the place of `MISSING1`? [2]

ii. What should go in the place of `MISSING2`? [3]

Question 3 This question is about determining if a number is the sum of two square numbers. A square number is the square of an integer. The number 0 and 1 are both square numbers.

(a) The following vector stores all square numbers from 1^2 to 7^2:

1	**4**	**9**	**16**	**25**	**36**	**49**
1	2	3	4	5	6	7

i. You are tasked with algorithmically searching this vector to see if it has an element with the value 1. Directly run through the binary search algorithm by hand on this vector, and return the index where the value is stored. Show explicitly each step taken in the algorithm. [7]

ii. Briefly explain why the vector above with the value 1 is an example of a worst-case input for the binary search algorithm? [3]

(b) One method to determine if an integer n is the sum of two square numbers is to try all combinations of the sum of two square numbers. Consider the following piece of incomplete pseudocode:

```
function SumSquares(n)
    if $n = 0$ then
        return TRUE
    end if
    for $0 \leq i \leq n$ do
        for $0 \leq j \leq n$ do
            if MISSING $= n$ then
                return TRUE
            end if
        end for
    end for
    return FALSE
end function
```

This function, when completed, should return TRUE if n is the sum of two square numbers, and FALSE otherwise.

i. What should go in the place of MISSING to complete this function? [3]

ii. What is the worst-case time complexity of this function? [4]

(c) Another method to solve the same problem as in part (b) of this question is to first create a vector of all square numbers from 0 to n^2 and then for each value i^2 stored in the vector, search the vector for the value $n - i^2$. Since the values in the vector will be sorted, the binary search algorithm can be used. Consider the following piece of pseudocode:

```
function BinarySquares(n)
    if $n = 0$ then
        return TRUE
    end if
    new Vector(n+1) squares
    for $0 \leq j \leq n$ do squares[j] $\leftarrow j^2$
    end for
    for $0 \leq i \leq n$ do
        if BinarySearch(squares, $n - i^2$) $=$ TRUE then
            return TRUE
        end if
    end for
    return FALSE
end function
```

This function will return TRUE if n is the sum of two squares, and FALSE otherwise. It assumes that the binary search algorithm is implemented by a function BinarySearch(vector, x), which returns TRUE if x is stored in the input vector, and FALSE otherwise.

i. Write the pseudocode function called BinarySearch(vector, x), which will implement the binary search algorithm. [9]

ii. Explain why the worst-case time complexity of implementing the function BinarySquares(n) is O(nlog n). [4]

END OF PAPER

.