UNIVERSITY OF LONDON

CM1035

BSc EXAMINATION

COMPUTER SCIENCE

Algorithms and Data Structures I

Release date: Thursday 10 March 2022 at 12:00 midday Greenwich Mean Time

Submission date: Friday 11 March 2022 by 12:00 midday Greenwich Mean 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 PDF to the appropriate area on the VLE. Your answers must have 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.

UL22/0098

SECTION A

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

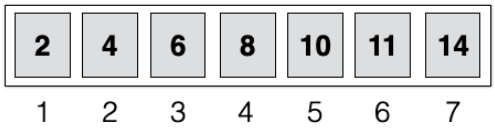
SECTION B

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

Question 2

This question is about searching sorted vectors of integers.

(a) Consider the following vector of integers:



By hand, directly run through the Binary Search algorithm on this vector searching for the value **9**. Show your working and how you choose elements to inspect.

(7 marks)

(b) An unsorted vector of integers of length **N** needs to be searched **N** times for a different integer value each time.

Approach A is to perform the Linear Search algorithm for each of the **N** integer values, and Approach B is to first sort the vector using Merge Sort and then perform the Binary Search algorithm for each of the **N** integer values. From the point of view of worst-case time complexity, which is the best approach, A or B? Explain your answer.

(6 marks)

(c) Consider the following piece of pseudocode:

```
function F1(vector)
    n ← LENGTH[vector]
    if n = 0 then
        return FALSE
    end if
    for 1 ≤ i ≤ n do
        for 1 ≤ j ≤ n do
            for 1 ≤ k ≤ n do
                if vector[i] + vector[j] + vector[k] = 0 then
                    return TRUE
                end if
            end for
        end for
    end for
    return FALSE
end function
```

What is the worst-case time complexity in terms of n of the algorithm described by the above pseudocode function F1, where the input is a vector of length n? Explain your answer.

(4 marks)

(d) The algorithm of F1 solves the following problem:

Given a vector of n integers, are there three integers in the vector that sum up to zero? The integers could be positive or negative, or zero.

Another algorithm to solve this problem is in the following pseudocode:

```
function F2(vector)
    n ← LENGTH[vector]
    if n = 0 then
        return FALSE
    end if
    vector ← SORT(vector)
    for 1 ≤ i ≤ n do
        for 1 ≤ j ≤ n do
            a ← 1
            b ← n
            while a ≤ b do
                c ← floor((a + b)/2)
                if vector[c] = − vector[i] − vector[j] then
                    return TRUE
                else if vector[c] < − vector[i] − vector[j] then
                    a ← c + 1
                else
                    b ← c − 1
                end if
            end while
        end for
    end for
    return FALSE
end function
```

The function SORT takes a vector of integers as input and returns the vector sorted in ascending order.

i. What is the worst-case time complexity in terms of n of the algorithm described by the above pseudocode function F1, where the input is a vector of length n? Assume that the function SORT implements Merge Sort. Explain your answer.

(4 marks)

ii. Instead of Merge Sort, suppose Quicksort is implemented by SORT instead. Briefly explain whether this affects the worst-case time complexity.

(3 marks)

(e) A generalisation of the problem above is to determine if, given a vector of n integers, are there one or more integers in the vector that sum up to zero.

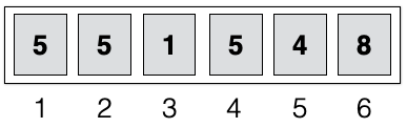
A colleague claims that they have an algorithm that can solve this problem with a worst-case time complexity that is a polynomial in n. Give and explain your opinion of the colleague's claim. You may use other resources, including the internet, as long as you appropriately reference these resources.

(6 marks)

Question 3

This question is about sorting vectors.

(a) Consider the following vector of integers:



You need to sort this vector by hand using an algorithm so that the smallest value is in the first element and the largest value is in the last element. **Showing your working** and how the vector changes in each step of the algorithm, implement the Insertion Sort algorithm on this vector.

(7 marks)

(b) Another sorting algorithm is Bubble Sort. From the point of view of worst-case time complexity, would you prefer to use Insertion Sort or Bubble Sort? Explain your answer giving the worst-case time complexities of both algorithms.

(4 marks)

(c) Another sorting algorithm is Merge Sort. From the point of view of time complexity, for certain input vectors it is advantageous to use Insertion Sort over Merge Sort. Give an example of a vector of length 5 where this is the case. Explain your answer.

(7 marks)

(d) Explain another reason why it would be preferable to use Bubble Sort rather than Merge Sort.

(4 marks)

(e) Consider the piece of JavaScript overleaf that implements Quicksort. Explain how, from an algorithmic perspective, the implementation of Quicksort here could be improved to minimise the resources required to run the code.

(8 marks)

```
function part(array, left, right){
      var lArray = [];
      var rArray = [];
      var mid = Math.floor((left + right) / 2);
      var pivot = array[mid];
      for (var i = left; i <= right; i++) {
            if (i !== mid) {
                  if (array[i] < pivot) {
                        lArray.push(array[i]);
                  } else {
                        rArray.push(array[i]);
                  }
            }
      }
      for (var i = 0; i < lArray.length; i++) {
            array[left + i] = lArray[i];
      }
      var final = left + i;
      array[final] = pivot;
      for (var i = 0; i < rArray.length; i++) {
            array[final + 1 + i] = rArray[i];
      }
      return final;
}

function sort(array, left, right) {
      if (right <= left) {
            return array;
      }
      var final = part(array, left, right);
      sort(array, left, final - 1);
      sort(array, final + 1, right);
      return array;
}
```

Question 4

This question is about stacks.

(a) Very briefly explain the advantage of using a linked list instead of an array to implement a stack.

(4 marks)

(b) Very briefly explain how a stack can be used to implement recursive functions.

(4 marks)

(c) Consider the following piece of JavaScript:

```
function Stack() {
   this.arr = [];
   this.push = function(item) {
         for (var i = this.arr.length - 1; i >= 0; i--) {
               this.arr[i + 1] = this.arr[i];
         }
         this.arr[0] = item;
   }
   this.pop = function(item) {
         if (this.arr.length == 0) {
               return "Stack underflow";
         }
         MISSING1
   }
   this.top = function() {
         return MISSING2
   }
   this.isEmpty = function() {
         MISSING3
   }
}
```

When completed the constructor `Stack` should implement a stack as an object where values are stored in a JavaScript array. The methods `push`, `pop`, `top` and `isEmpty` implement the push!, pop!, top and empty? operations. Answer the following:

i. What should replace `MISSING1` in this code?

(3 marks)

ii. What should replace `MISSING2` in this code?

(2 marks)

iii. What should replace `MISSING3` in this code?

(3 marks)

(d) Consider the following piece of JavaScript:

```
function stacksEqual(stack1, stack2) {
   while (!stack1.isEmpty() && !stack2.isEmpty()) {
        if (stack1 !== stack2) {
             return false;
        }
        stack1.pop();
        stack2.pop();
   }
   return stack1.isEmpty() && stack2.isEmpty();
}
```

The function `stacksEqual` takes two objects `stack1` and `stack2`, which are implementations of stacks. The function should return true if the two objects have exactly the same elements with the same values, and false otherwise. However, this function does not work correctly. Explain why this function will not work as desired, and explain any other problems with this approach.

(7 marks)

(e) A colleague has claimed that the most efficient method for implementing a stack in JavaScript is to use a JavaScript array, just as in part (c) of this question. Do you agree or disagree with your colleague? Explain your answer.

You may use other resources, including the internet, as long as you appropriately reference these resources.

(7 marks)

END OF PAPER