THIS PAPER IS NOT TO BE REMOVED FROM THE EXAMINATION HALL

UNIVERSITY OF LONDON

CM1035

BSc EXAMINATION

COMPUTER SCIENCE

Algorithms and Data Structures I

Thursday 5 March 2020 : 10.00 – 12.00

Time allowed: 2 hours

DO NOT TURN OVER UNTIL TOLD TO BEGIN

INSTRUCTIONS TO CANDIDATES:

This examination paper is in two parts: Part A and Part B. You should answer **ALL** of question 1 in Part A and **TWO** questions from Part B. Part A carries 40 marks, and each question from Part B carries 30 marks. If you answer more than **TWO** questions from **Part B** only your first **TWO** answers will be marked.

All answers must be written in the answer books; answers written on the question paper will not be marked. You may write notes in your answer book. Any notes or additional answers in the answer book(s) should be crossed out.

The marks for each part of a question are indicated at the end of the part in [.] brackets. There are 100 marks available on this paper.

Calculators are not permitted in this examination.

PART A

Candidates should answer **ALL** of Question 1 in Part A.

Question 1 If instructed to choose only one option, do not choose multiple options. If you select multiple options, your answer will be marked as 0. If otherwise instructed to select all options that apply, there will be multiple correct options, but you can lose marks for selecting incorrect options.

(a) Consider the following problem:

Given a positive integer n, is n even?

Which of the following is the type of the expected output for this problem? Choose only one option. [3]

i. Boolean

ii. Integer

iii. Number

iv. None of the above.

(b) Consider the following piece of pseudocode:

```
1: a ← 2
2: a ← 2a
3: function EQUALTO(n)
4:     if n=5 then
5:         return n
6:     end if
7:     return n+1
8: end function
9: a ← EQUALTO(a)
10: a ← a^2
11: a← EQUALTO(a)
```

i. What is the value of a at the end of line 2? [1]

ii. What is returned by EQUALTO(6)? [1]

iii. What is the value of a at the end of line 9? [1]

iv. What is the value of a at the end of line 11? [1]

(c) From where is an element removed from a queue? Choose only one option. [3]

i. tail

ii. top

iii. head

iv. index

(d) Which of the following describes the operation that adds a new element with the value o to a queue? Choose only one option. [3]

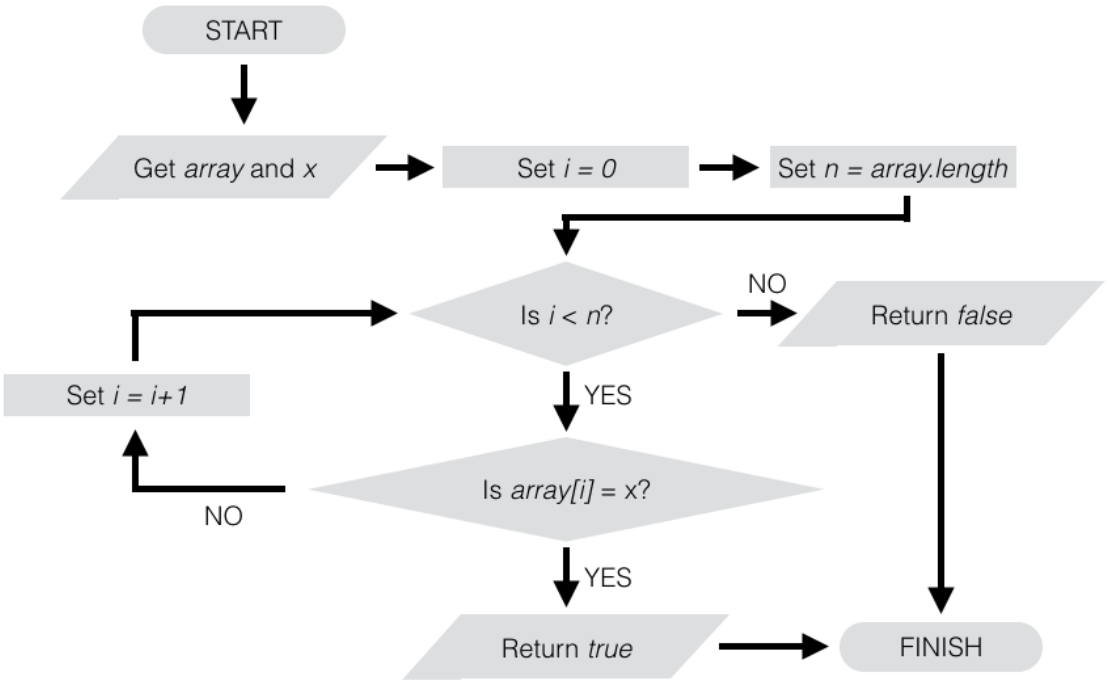
i. dequeue![o]

ii. push![o]

iii. pop![o]

iv. enqueue![o]

(e) Consider the following flowchart:



Which of the following is the name of the algorithm being implemented here? Choose only one option. [2]

i. Binary search

ii. Linear search

iii. None of the above.

(f) Consider the following incomplete piece of JavaScript:

```
1 function linear1(array,x) {
2   n = MISSING1;
3   for (var i = 0; i < n; i++) {
4     if (array[i] == x) {
5       return MISSING2;
6     }
7   }
8   return false;
9 }
```

When completed, this piece of JavaScript code will implement the algorithm implemented in question 1(e). Which of the following should go in the place of `MISSING1`? [2]

i. `array[0]`

ii. `array.length`

iii. `array.length()`

iv. `array.length - 1`

(g) Which of the following should go in the place of `MISSING2`? [2]

i. `true`

ii. `false`

iii. `i`

iv. `array[i]`

(h) For which of the following reasons is it preferable to use a linked list to implement a dynamic array rather than use an array data structure? (note that this is distinct from a JavaScript array) Choose all options that apply. [4]

i. Linked lists are linear, but arrays are not

ii. Element of data can be easily added and removed from a linked list, but not from an array

iii. Linked lists cannot have data easily added and removed, just like dynamic arrays

iv. Elements can be easily removed from arrays, but cannot be removed from linked lists

v. The size of dynamic arrays can vary, and the size of an array cannot

vi. Linked lists do not store pointers, whereas arrays do

(i) Which of the following is the worst-case time complexity in 'Big O' notation of the binary search algorithm in n for a sorted vector of length n? Choose only one option. [4]

i. O(1)

ii. O(log n)

iii. O(n^2)

iv. O(n log n)

(j) Consider the following piece of incomplete pseudocode for a function FACTORIAL(n), which takes the number n as an input parameter, and should return the factorial of n. Recall that the factorial is the product of all numbers from 1 to n with the factorial of 0 being equal to 1.

```
function FACTORIAL(n)
    if n = 0 then
        return 1
    end if
    MISSING
end function
```

Which of the following should replace MISSING to complete this recursive function? Choose only one option. [4]

i. **return** FACTORIAL(n)

ii. **return** n $\times$ FACTORIAL(n)

iii. **return** n $\times$FACTORIAL(n-1)

iv. **return** FACTORIAL(n - 1)

(k) You are trying to perform a task on a large database of n elements, where n will tend to get very large. One colleague tells you that they have an algorithm (called algorithm **A**) that takes at most n^3+2n^2+100 time-steps, another colleague says they found an algorithm (called algorithm **B**) that takes at most $0.01(2^n+2n)$ steps.

i. Which of the following is the worst-case time complexity of algorithm **A**? Choose only one option. [2]

- O(n^2)
- O(n)
- O(n^3)

ii. Which of the following is the worst-case time complexity of algorithm **B**? Choose only one option. [2]

- O(n^2)
- O(2^n)
- O(n)

iii. From the point of view of worst-case time complexity, which of the two algorithms should you pick to complete this task? [1]

(l) Which of the following algorithms are 'divide and conquer' algorithms but not 'decrease and conquer' algorithms? Select all options that apply. [4]

i. Merge sort

ii. Bubble sort

iii. Quicksort

iv. Linear search

v. Insertion sort

vi. Binary search.

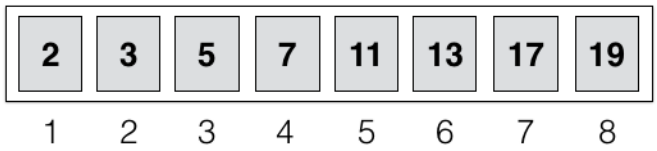
PART B

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

Question 2

In this question, the objective is to design an algorithm to determine if a number is prime. That is, given a positive integer as input, whether it can be perfectly divided (without remainders) by any other integer other than itself or 1.

(a) Consider the following vector of numbers called PRIMES:



This vector contains the first eight prime numbers, with the relevant index indicated below the elements. We will search this vector to determine if the number **18** is a prime number. You are tasked with algorithmically searching this vector for the number **18**. Directly run through the binary search algorithm on this vector. Show explicitly the steps taken in the algorithm: make sure you indicate whether **18** is found, or otherwise. [7]

(b) Does your result from question 2(a) mean that **18** is, or is not, a prime number? [1]

(c) Briefly explain why, considering the characteristics of the vector, it was preferable in question 2(a) to implement the binary search algorithm on the vector PRIMES instead of the linear search algorithm. [2]

(d) If given an arbitrary positive integer n, one method to decide if n is a prime number is to try every integer j from 2 to $n-1$ to see if n is perfectly divisible by j: if n is perfectly divisible by any of the values of j, then it is not prime.

Consider the following incomplete pseudocode function that should describe this algorithm for determining if a number if prime.

```
1:  function ISPRIME(n)
2:      if n<2 then
3:          return FALSE
4:      end if
5:      for MISSING1 do
6:          if MISSING2 then
7:              return FALSE
8:          end if
9:      end for
10:     return TRUE
11: end function
```

i. What should go in the place of MISSING1? [4]

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

(e) An improvement over the previous approach in question 2(d) is to range j from 2 to $\lfloor\sqrt{n}\rfloor$ since any number that is not prime can be written as $j \times k$, so j will be at most $\lfloor\sqrt{n}\rfloor$ (and $\lfloor x\rfloor$ is the floor of x).

i. What is the corresponding completed line of JavaScript code for line 6 of the pseudocode in question 2(d)? [3]

ii. What is the corresponding completed line of JavaScript code for line 5 of the pseudocode in question 2(d) where now we use the improved version mentioned above? [4]

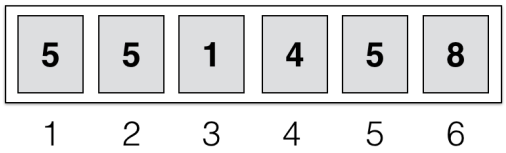
(f) Give an argument for why the worst-case time complexity of the algorithm implemented in question 2(e) is $O(\sqrt{n})$. [3]

(g) We define the decision problem PRIMES as the set of all positive integers that are prime numbers. Briefly outline why the time complexity in question 2(f) imply that PRIMES is in the complexity class **EXP**? HINT: Remember that an integer n requires $O(\log n)$ digits to be stored as an input. [3]

Question 3

In this question, we will consider sorting algorithms.

(a) Consider the following vector of integers:



Implement the insertion sort algorithm on the vector to sort it from smallest (in the first element) to largest, explicitly showing how the vector changes in the algorithm. [7]

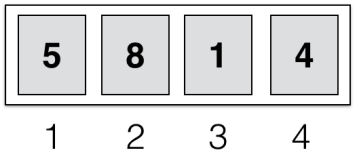
(b) What is the worst-case time complexity in n of the insertion sort algorithm for a vector of length n? [3]

(c) 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. [3]

(d) Write an example for a worst-case input vector of length 6 for the bubble sort algorithm. [4]

(e) Explain why your example in question 3(d) is an example of a worst-case input, making reference to worst-case time complexity of bubble sort. [4]

(f) Consider the following vector of integers:



Implement the quicksort algorithm on the vector to sort it from smallest (in the first element) to largest, explicitly showing how the vector changes in the algorithm. [6]

(g) From the point of view of worst-case space complexity, would you prefer to use insertion sort or quicksort? Explain your answer. [3]

Question 4

In this question we will consider stacks, concrete data structures and recursion.

(a) A stack is an abstract data structure with several operations. Describe each of the following operations on the stack:

i. Push![o] [3]

ii. Pop! [2]

iii. Top! [2]

(b) Apart from the operations in question 4(a), what is the other operation that can be performed on a stack? Describe this operation in terms of what is returned when it is performed. [3]

(c) Consider the following piece of pseudocode:

```
new Stack s
PUSH[1,s]
PUSH[3,s]
POP[s]
x←TOP[s]
PUSH[2+x,s]
PUSH[5+x,s]
POP[s]
x←TOP[s]
```

i. What is final value of x at the end of this pseudocode? [2]

ii. To implement the operations in this pseudocode we will use an array data structure. If we initially have an array of two elements, both empty, how many more new arrays do we need to create in total for this implementation? [2]

iii. The stack may also be implemented as a linked list. Draw a linked list of the final form of the stack *s* at the end of the pseudocode, with the top of the stack being the first element of the list. [5]

(d) Consider the following piece of pseudocode, which implements the function of taking the sum of all integers from 1 to n for a given positive integer n.

```
function SUM(n)
    x←0
    for 0 ≤ i ≤n do
        x←x+ i
    end for
    return x
end function
```

i. Give an argument why the worst-case time complexity of the function SUM(n) is $O(\mathrm{n})$ for input integer n. [3]

ii. Write a pseudocode function called RECSUM(n), which is a recursive algorithm that calculates the same thing as SUM(n). It also takes n as input parameter and produces the same output. Do not use any form of iteration. [4]

iii. Give an argument why the worst-case time complexity of your function RECSUM(n) should be $O(\mathrm{n})$ for input integer n. Note: remember the data structure used to handle recursive function calls. [4]

END OF PAPER

.