CM1035

BSc EXAMINATION

COMPUTER SCIENCE

Algorithms and Data Structures I

Release date: Wednesday 7 September 2022 at 12:00 midday British Summer Time

Submission date: Thursday 8 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 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

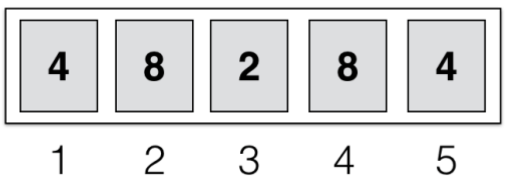
Candidates should answer the **TEN** Multiple Choice Questions (MCQs) quiz, **Question 1** in Section A on the VLE.

SECTION B

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

Question 2 This question is about checking if a vector is a palindrome. A vector is a palindrome if it is 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) Given 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 3

This question is about searching and solving a problem in modular arithmetic.

(a) Consider the following vector of integers:

2	**3**	**5**	**7**	**11**	**13**	**17**
1	2	3	4	5	6	7

By hand, directly run through the Binary Search algorithm on this vector searching for the value **5**. Show your working and how you choose elements to inspect. [7]

(b) It can be argued that the Binary Search algorithm is optimal. Very briefly explain what this means. You do not have to provide an argument, just explain what the statement means. [3]

(c) A modular square root of a non-negative integer *y* modulo *N* is a non-negative integer *x* such that $x^2 \equiv y \pmod{N}$, i.e. x^2 and *y* are congruent modulo *N*. For instance, if *y* is 2 and *N* is 7, then *x* could be 3 since 9 mod 7 is 2, or *x* could be 4 since 16 mod 7 is also 2. That is, there can be multiple square roots. (There is a short revision on modular arithmetic at the end of this question.) The modular square roots *x* and *y* are also assumed to be integers less than *N*. Note that *N* is always assumed to be greater than 0.

Consider the following piece of pseudocode that finds the smallest square root of an integer *y* modulo *N*:

```
function RecMod(a, N)
    if $a < N$ then
        return a
    end if
    return RecMod($a - N$, N)
end function

function ModSquareRoot(y, N)
    for $0 \le x \le N$ do
        if RecMod($x^2$, N) $=$ y then
            return x
        end if
    end for
end function
```

i. Briefly explain why the worst-case time complexity of RecMod(*a, N*) is *O(a/N)* for inputs *a* and *N.* [5]

ii. What is the worst-case time complexity of MODSQUAREROOT(*y, N*) in Big O notation and in terms of the inputs *N* and/or *y*? [2]

iii. Briefly explain your solution to part 2 of (c), i.e. the worst-case time complexity of MODSQUAREROOT(*y, N*). [5]

(d) Consider the following piece of pseudocode that claims to find the smallest square root of an integer *y* modulo *N*:

```
function ModNew(a, N)
    r ← $\lfloor a/N \rfloor$
    return a − (r×N)
end function

function ModRootNew(y, N)
    a ← 0
    b ← N
    while a ≤ b do
        q ← $\lfloor N/2 \rfloor$
        if ModNew($q^2$, N) = y then
            return q
        else if ModNew($q^2$, N) < y then
            a ← q + 1
        else
            b ← q − 1
        end if
    end while
end function
```

If this algorithm worked, in general, it would give an algorithm that could find the smallest modular square root of an integer *y* in time that is *O(log N)* for integer input *N*. Explain why this algorithm cannot work. You can use the internet to perform research for your answer, as long as you appropriately reference the materials used. [8]

Short Modular Arithmetic Revision

A non-negative integer is an integer greater than or equal to 0. Every non-negative integer *a* can be written as $a = pN + q$ where *p*, *q* and *N* are all non-negative integers and $N > 0$, $q < N$. In modular arithmetic modulo *N* we restrict to arithmetic over non-negative integers between (and including) 0 and $N – 1$. For every non-negative integer *a*, the integer *a* (mod *N*) is exactly equal to the value q above, i.e. once we delete the part of *a* that can be perfectly divided by *N*, which is *pN*, we are left with *q*. For instance, 17 (mod 4) is equal to 1 since 17 = 4 x 4 + 1. We never take *N* to be zero since we cannot divide a number by 0.

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]

(b) Very briefly explain how a stack can be used to implement recursive functions. [4]

(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 push, pop, top and empty operations. Answer the following:

i. What should replace MISSING1 in this code? [3]
ii. What should replace MISSING2 in this code? [2]
iii. What should replace MISSING3 in this code? [3]

(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 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]

(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. [7]

END OF PAPER