School of Computing and Information Systems
COMP20007 Design of Algorithms
Semester 1, 2020
Mid Semester Test

Instructions to Students

- The total time to read, complete, scan and upload your solutions to this test is 1 hour. You should allow at least 15 minutes to scan and upload your solutions.
- This test contains 4 questions which will be marked out of 10 marks but will not contribute to your final grade.
- This is an open book test. You may consult resources made available via the LMS or the text book.
- You must not communicate with other students or make use of the Internet.
- Solutions must be written on separate pieces of paper.
- You must write your solution to each question on a new sheet of paper and clearly indicate which question you are answering on each page.
- You must use a Scanner app on your smartphone to scan your solutions, email them to your laptop and then submit a PDF file to Gradescope via Canvas.
- A Gradescope guide to scanning your test can be found here:

  `https://gradescope-static-assets.s3-us-west-2.amazonaws.com/help/submitting_hw_guide.pdf`
- The Dropbox and Microsoft OneDrive mobile apps also have scanning capabilities.

Question 1 [2 Marks]

We know, from lectures, the following facts. For $0 < \epsilon < 1 < c$,

$$1 \prec \log n \prec n^{\epsilon} \prec n^{c} \prec n^{\log n} \prec c^{n} \prec n^{n},$$

where $f(n) \prec g(n)$ means both $f(n) \in O(g(n))$ and $g(n) \notin O(f(n))$ (i.e. $g(n)$ is not in $O(f(n))$).

For the following pairs of functions, $f(n), g(n)$, determine if $f(n) \in \Theta(g(n)), f(n) \in O(g(n))$, or $f(n) \in \Omega(g(n))$, making the strongest statement possible. That is, if both $f(n) \in O(g(n))$ and $f(n) \in \Omega(g(n))$ are true, you **must** answer with the form $f(n) \in \Theta(g(n))$. You must answer with $f(n)$ on the left and $g(n)$ on the right, for example, you may not answer $g(n) \in O(f(n))$.

You **must** show **all** working. A correct answer that does not show your working will result in 0 marks.

(a) $f(n) = (n+1)^3$, $g(n) = (2n)^3$

(b) $f(n) = 3^{n+1}$, $g(n) = (3+1)^n$

(c) $f(n) = n^3 + 1.1^n$, $g(n) = (n^3)^2 + 1.1^n$

(d) $f(n) = \log(n^n)$, $g(n) = \sqrt{n}$

Question 2 [2 Marks]

A string of characters is called a palindrome if reading the characters from left to right gives the same sequence as reading the characters from right to left.

For example: `abccba`, `ahgiigha` and `abgllggllgba` are all palindromes while `abcdabcd`, `lghddgl` and `abcd` are not.

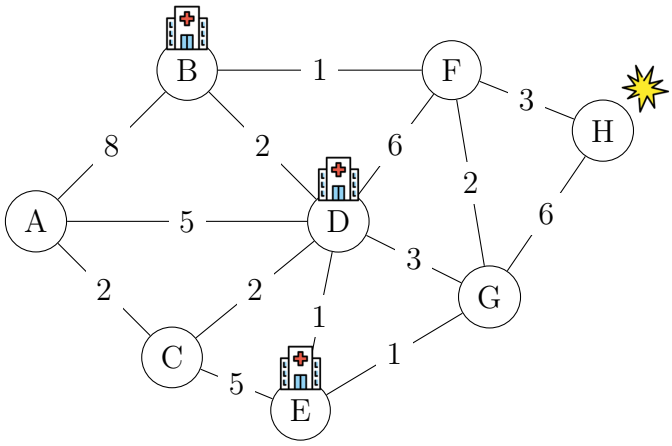
Write the pseudocode for an algorithm which, given a string and its length n, can identify a palindrome **using only one scan from left to right** (which means you can only access each letter once). Your algorithm **must employ the use of either a queue or a stack**, and you **must justify your choice**.

If you use a stack myou must use PUSH() and POP(), and if you use a queue you must use ENQUEUE() and DEQUEUE().

Assume for simplicity every input will be of even length.

Question 3 [3 Marks]

The following graph represents a road network where each edge presents a road segment. The costs of the road segments are given in the figure below. There are three hospitals at nodes B, D, and E, and an accident occurs at node H.



(a) Which algorithm would you use to find the nearest hospital from the accident. Explain how and why you would use this algorithm.

(b) Which is the nearest hospital, which path would you take and what is the cost of this path? You must show the execution of the algorithm described above.

(c) List the order of the nodes visited from H in the graph when a breadth-first search is used.

You should break ties in alphabetic order.

Question 4 [3 marks]

Consider the following recursive function, which takes an unordered array of integers, A, and an integer, k, and returns TRUE if k appears in A and FALSE otherwise.

```
function ThirdsSearch(A[0..n-1], k)
    if n == 0 then
        return FALSE
    else if n == 1 then
        return (A[0] == k)
    else
        a ← ThirdsSearch(A[0..n/3 − 1], k)
        b ← ThirdsSearch(A[n/3..n × 2/3 − 1], k)
        c ← ThirdsSearch(A[n × 2/3..n − 1], k)
        return (a OR b OR c)
```

For simplicity, you may assume the input array is of length $n = 3^m$ for some positive integer m, so that each recursive call gets an exact third of the input array.

(a) Write down, and **solve**, a recursive formula $W(n) = \ldots$ which describes the number of steps taken by ThirdsSearch on an array of length n, for the worst case input. Make sure you include the base case(s). Keep in mind that the basic operation is usually the most frequent or most expensive operation. Remember the formula for the geometric series states that

$$\sum_{i=0}^{k} x^i = \frac{x^k - 1}{x - 1}.$$

(b) Use the language of Ω, O, Θ to bound the time complexity of ThirdsSearch, using $T(n)$ for its runtime. You must include an upper **and** lower bound. Full marks are awarded only to the tightest possible bounds.

(c) Explain how ThirdsSearch could be slightly modified to improve its efficiency in the best case, and repeat the analysis from part (b), with justification.

END OF TEST