The University of Nottingham

SCHOOL OF COMPUTER SCIENCE

A LEVEL 4 MODULE, AUTUMN SEMESTER 2018-2019

ADVANCED ALGORITHMS AND DATA STRUCTURES

Time allowed: TWO HOURS

Answer ALL FOUR questions

Marks available for sections of questions are shown in brackets in the right-hand margin.

No calculators are permitted in this examination.

Question 1. [25 marks total]
Complexity classes, big-O notation, recurrence relations

(a) [8 marks]
Here are a list of functions and a list of complexity classes:

$f_1 = 2n - 5$	i.	$O(n)$
$f_2 = \log(n)^3$	ii.	$\Omega(n)$
$f_3 = 2^n$	iii.	$\Theta(n)$
$f_4 = n^3$	iv.	$o(n)$

For each complexity class, state which of the functions are members of the class (there may be several for each class).

(b) [8 marks]
Consider the following code fragment, defining a function **strange** on lists:

```
strange ($l$) :
  $n$ = length ($l$)
  if $n < 3$
    then return (1)
    else
      $k = \lfloor n/3 \rfloor$
      $r_1$ = strange ($l[0 .. (2k-1)]$)
      $r_2$ = strange ($l[k .. (n-1)]$)
      return (fuse($r_1, r_2$))
```

Assuming **fuse**(r_1, r_2) runs in $O(n)$, write a recursion relation expressing the running time $T(n)$ of **strange**.

(c) [9 marks]
Determine the exact asymptotic complexity of $T(n)$, using the Θ notation. (You can use the Master Theorem.)

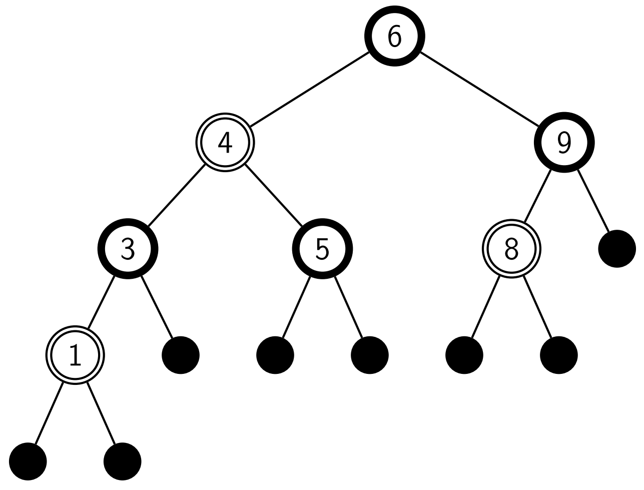
Question 2. [25 marks total]
Red-Black Trees

(a) [8 marks]
Explain what *dynamic-sets* are and what basic operations must be provided by an implementation of them?

Give a definition of *binary search trees* and explain informally how they implement dynamic sets and why they are potentially more efficient than a list implementation.

(b) [8 marks]
Give the definition of *Red-Black Trees*. Clearly explain what the data structure is and which properties it must satisfy.

(c) [9 marks]
Consider the following red-black tree t (thick circle = black node, double circle = red node):



Draw the red-black tree t_1 obtained by inserting the element 2 into it (and rebalancing): $t_1 = \textbf{insert}\, 2\, t$.

Draw the red-black tree t_2 obtained by deleting the element 5 from the previously obtained tree: $t_2 = \textbf{delete}\, 5\, t_1$.

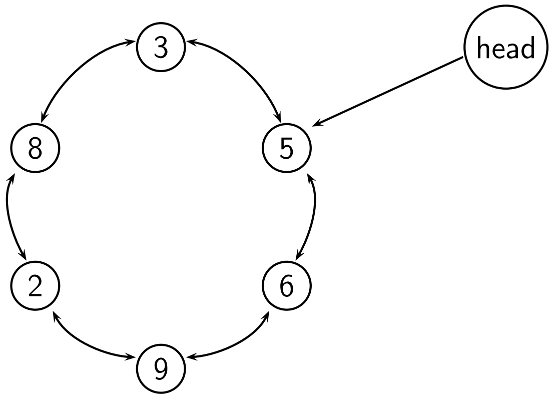
Question 3. [25 marks total]
Amortized analysis and Wheels

(a) [8 marks]
What do we mean by the *amortized complexity* of an algorithm? Describe the *potential* method of amortized analysis of time complexity works.

(b) [8 marks]
Wheels (doubly-linked circular lists) are data structures containing a number of elements placed around the edges of a circle, with a reading head pointing to one of the elements. We have operations to read the head element, move the head left (anti-clockwise) or right (clockwise), to add an element to the left of the head and to delete the head. Suppose that the wheels are represented as pairs of (simply-linked) lists, both lists should be non-empty if the wheel contains at least two elements.

type Wheel a = ([a],[a])}

The actual wheel corresponding to a pair (f,r) has the elements of *f* under and to the right of the head and the elements of *r* to the left of the head. For example the wheel



can be represented by the pair of lists $([5, 6, 9], [3, 8, 2])$, or, equivalently, by $([5, 6, 9, 2], [3, 8])$.

Given this data structure, define (in pseudo-code) efficient operations to read the head element, moving the head left and right, inserting and deleting elements:

```
head :: Wheel a -> a
left :: Wheel a -> Wheel a
right :: Wheel a -> Wheel a
insert :: a -> Wheel a -> Wheel a
delete :: Wheel a -> Wheel a
```

Be careful to maintain the invariant that if the wheel contains at least two elements, then both lists must be non-empty. For example:

```
left ([5,6,9,2,8],[3]) = ([3,5,6],[9,2,8])
```

(c) [9 marks]
Define a potential function on wheels. (Hint: use the difference in length of the two list components).

Use it to prove that all operations have amortized time cost of $O(1)$.

Question 4. [25 marks total]
Dynamic Programming, Graph Algorithms, Maximum Flow

(a) [8 marks]
In Dynamic Programming, what does it mean that a problem has *optimal substructure* and *overlapping substructure*? Why do these properties make the problem suitable for DP? Illustrate your answer with the problem of optimal rod cutting.

(b) [8 marks]
Define the two different representations of a directed graph: as *adjacency list* and as *adjacency matrix*.

Give an informal definition of the *breadth-first search* algorithm to compute the minimum distance of each vertex from a source s.

(c) [9 marks]
What is the definition of a flow function in a flow network? What is the definition of the value of a flow?

Also, briefly describe the algorithm used to solve the maximum flow problem. In particular, define *residual network* and *augmenting path*.