The University of Nottingham

SCHOOL OF COMPUTER SCIENCE

A LEVEL 2 MODULE, FULL YEAR, 2022-2023

ALGORITHMS CORRECTNESS AND EFFICIENCY

Time allowed TWO HOURS

Candidates may complete the front cover of their answer book and sign their desk card but must NOT write anything else until the start of the examination period is announced

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.

Dictionaries are not allowed with one exception. Those whose first language is not English may use a standard translation dictionary to translate between that language and English provided that neither language is the subject of this examination. Subject specific translation dictionaries are not permitted.

No electronic devices capable of storing and retrieving text, including electronic dictionaries, may be used.

DO NOT turn your examination paper over until instructed to do so

INFORMATION FOR INVIGILATORS: Students should write their answers to these questions in the answer book. The exam paper should be collected and placed inside the answer book.

1) This question is concerned with Formal Reasoning in Lean. (25 marks total)

a) Which of the following propositions of propositional logic are provable in Lean? (Assuming P,Q : Prop)

i) P ↔ P ∨ P
ii) P ∧ Q ↔ P ∨ Q
iii) ¬ (P ∨ ¬ P)
iv) ¬ (P ∧ ¬ P)
v) (P ↔ ¬ P)

Answer (yes/no) for each proposition.

(10 marks)

b) Which of the following propositions of predicate logic are provable in Lean? (Assuming A : Type, PP : A → Prop, Q : Prop)

i) ∀ x : A, (PP x ↔ PP x)
ii) ∃ x : A, (PP x ↔ PP x)
iii) ∃ x : A, (PP x ∧ Q) ↔ (∃ x : A, PP x) ∧ Q
iv) (∃ x : A, PP x ∨ Q) ↔ (∃ x : A, PP x) ∨ Q
v) ¬ (∀ x:A, false)

Answer (yes/no) for each proposition. (10 marks)

c) Given QQ : ℕ → Prop , how would you prove the principle of induction in Lean?

```
theorem ind : QQ 0 →
  (∀ n : ℕ, QQ n → QQ (succ n))
  → ∀ n:ℕ, QQ n :=
begin
  sorry,
end
```

Replace “sorry” by a valid Lean proof. (5 marks)

2) This question concerns the 'big-Oh' family and recurrence relations (25 marks total)

(a) Consider the function

$$f(n) = \begin{cases} 3\ n + n^2 & \text{if n is odd} \\ 2\ n & \text{otherwise (i.e. if n is even)} \end{cases}$$

From the definitions:

i) Prove or disprove that $f(n)$ is $O(n^2)$ (`Big-Oh of n squared').

ii) Prove or disprove that $f(n)$ is $\Omega(n^2)$ (`Big-Omega of n squared').

(9 marks)

(b) The usual definition of 'little-oh' is that

```
f(n) is o( g(n) ) if and only if
forall  c > 0 such that,
exists n ≥ n0
   f(n) < c g(n)
```

Consider the following (incorrect) definition, of "useless little oh" "ulo":

```
f(n) is ulo( g(n) ) if and only if
exists c > 0 such that,
exists n ≥ n0
   f(n) < c g(n)
```

Answer the following two questions (show your working)

i) From the definition, prove or disprove that n is o(n)

ii) From the definition, prove or disprove that n is ulo(n)

Briefly explain, using your two results, why 'ulo' would not be a useful definition, to be added to the Big-Oh family, for describing the scaling behaviours of algorithms.

(9 marks)

(c) Consider the recurrence relation

$$T(n) = 2\,T(n/2) + d \qquad \text{with} \quad T(1) = 1$$

where d is some arbitrary constant.

Answer the following, showing and explaining your working:

i) Solve the relation exactly; that is, give a formula for T(n).

ii) Prove your answer from i) is correct using induction.

iii) Find the Big-Theta behaviour of T(n) using the Master theorem. Show that it agrees with the answer from your exact solution.

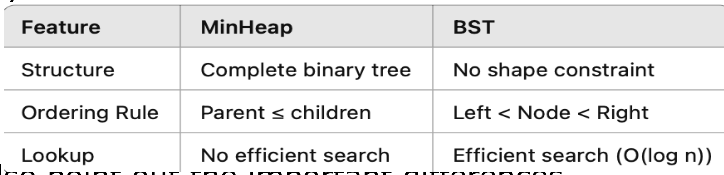
(7 marks)

3) This question concerns Heaps and Hash tables (25 marks total)

(a) Describe and explain the conditions for a Binary tree to be a

i) Heap (specifically, a 'minHeap')

ii) Binary Search Tree (BST)



Feature	MinHeap	BST
Structure	Complete binary tree	No shape constraint
Ordering Rule	Parent ≤ children	Left < Node < Right
Lookup	No efficient search	Efficient search (O(log n))

Briefly justify the definitions/conditions, and also point out the important differences.

(7 marks)

(b) Describe and explain how to store a binary minHeap within an array – i.e. as an "array-based heap". Explain how the operations of adding an element to the heap and the operation of removing the minimum heap work in the context of the "array-based heap".

(8 marks)

(c) Suppose that it is necessary to extend the array-based heap with a method "removeKey(int k)" that will remove the key k from the heap. For simplicity, you can assume that the keys are unique within the heap. That is, the heap contains at most one copy of the key k, but note that the key k might not appear at all.

Give and explain an algorithm to do this O(n) time.

(6 marks)

(d) Consider the situation of part (c) but now it is desired to do the removal on O(log n) time.

To be able to do this it is proposed to use a hash table, based on the keys in the heap, and storing the index in the array at which the key is stored.

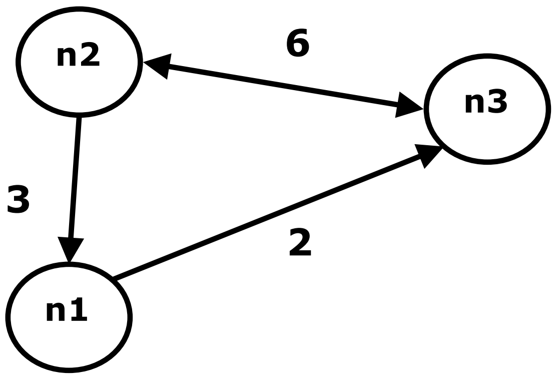
Describe and explain a method to do this. The method should usually work in O(log n) time. Explain why the use of the hash table does not allow to guarantee that it always work in O(log n) time.

(4 marks)

4) This question concerns dynamic programming methods. (25 marks total)

(a) Explain, and give pseudo-code for, the Floyd-Warshall algorithm to find the lengths of the shortest paths between each pair of vertices in a graph. The graph can have a mix of directed and bidirectional edges, with distance d(i, j) from node i to node j. You can assume the distances are such that there are no negative cycles.

Then use the algorithm on the (directed) graph below to find the matrix of distances for all-pairs of shortest paths. Show your working.



(12 marks)

(b) Suppose that you are given a set S of "coins", with values that are positive integers, and a target amount of change C to give to a customer. Give a dynamic programming algorithm to determine whether or not it is possible to give exactly C in change using a subset of the coins in S. Consider the example S = { 1, 3, 7 } it is possible give change of C=4, but not C=9.

Show how the algorithm works for the case C=9.

(8 marks)

(c) As a variant of the problem in (b) suppose that the elements of the set S are "stretch factors" for a machine that takes a metal rod and stretches it (or rolls it out) so that the length of the rod is multiplied by the stretch factor. The rod initially starts as length 1 and the question is whether a target length L is achievable using a subset of the machines, but each machine can be used only once. Consider the example S = { 2, 3, 5 }, then it is possible give an overall final length of L=6, using the stretches of 2 and 3, but not L=12.

Show how to modify the algorithm of (b) to handle multiplication of the stretch factors – instead of the addition of the coins.

(5 marks)

END