- This exam has 6 questions for a total of 135 marks, of which you can score at most 100 marks.
- You may answer any subset of questions or parts of questions. All answers will be evaluated.
- Go through all the questions once before you start writing your answers.
- Use a pen to write. Answers written with a pencil will *not* be evaluated.
- Warning: CMI's academic policy regarding cheating applies to this exam.

You do *not* have to use loop invariants for proving the correctness of algorithms; but you must correctly explain why each loop (if there are some) does what you expect it to do. Of course, you *may* use loop invariants if you wish.

Unstated assumptions and lack of clarity in solutions can and will be used against you during evaluation. You may freely refer to statements from the lectures in your arguments. You don't need to reprove these unless the question explicitly asks you to, but you must be precise.

Please ask the invigilators if you have questions about the questions.

1. In this question we consider graphs that are finite and undirected. Such a graph is *simple* if has neither multiple edges nor self-loops. An *isolated* vertex in a graph is a vertex with no edge incident on it. We say that a graph is *special* if it is simple and has no isolated vertex. The notation $|S|$ denotes the cardinality of set S.

   Consider the following claim and its proof:

   Claim

   Every special graph $G = (V, E)$ that satisfies $|V| = |E| + 1 \geq 2$, is acyclic.

   Proof: By induction on $|V|$.

   - Base case: $|V| = 2, |E| = 1$: correct by inspection.
   - Inductive assumption: suppose the claim holds for all special graphs with at most k vertices, for some $k \geq 2$.
   - Inductive step: Let $G = (V, E)$ be a special graph with $|V| = k = |E| + 1$. Consider an arbitrary graph G' obtained from G by adding (i) a new vertex v, and (ii) an arbitrary edge from v to some vertex of G. Then G' (i) has $k+1$ vertices and k edges, (ii) has no isolated vertices, and (iii) is acyclic. Hence proved.

(a) Provide a counter-example to the claim, with at most 10 vertices. [5]

(b) Clearly explain what is wrong with the above proof. [5]

2. (a) Write the pseudocode for a *recursive* algorithm QUOTREM(x, y) that takes two integers $x \geq 0, y \geq 1$ as arguments, and uses *repeated subtraction* to find the quotient q and the remainder r that are obtained when x is divided by y. [10]

   Make sure that your algorithm handles the base case(s) correctly. You will get the credit for this part only if your algorithm is (i) recursive, and (ii) correct.

   (b) Prove using induction that your algorithm of part (a) is correct. Make sure that you handle (all) the base case(s). [10]

3. The SecondBest problem is defined as follows:

SecondBest

- Input: An integer $n \geq 1$ and an array A of n integers. Array A is indexed from 0; its elements are thus $A[0], A[1], \ldots, A[(n-1)]$.
- Output: The *second largest* number x which is present in A, or the special value NIL if there is no such number in A.

  A number x is the second largest number in A if: (i) x is present in A; (ii) the largest number present in A is y, and $y > x$ holds; and, (iii) there is no element z in A such that $x < z < y$ holds.

   (a) Give an example of an input to this problem with $n = 5$, where the (correct!) output value is NIL. [5]

   (b) Write the *complete* pseudocode for an algorithm SECONDBEST(n, A) that solves the SecondBest problem in $\mathcal{O}(n)$ time in the worst case. You will get the credit for this part only if your algorithm is correct, and runs within the required time bound. [10]

   (c) Argue that your algorithm of part (a) correctly solves the problem. [10]

   (d) Prove that your algorithm from part (a) runs in $\mathcal{O}(n)$ time in the worst case. For this you may assume that each array operation, and each comparison of a pair of numbers, take constant time. Clearly state any other assumptions that you make. [10]

4. Prove the following statement using the definition of the $\mathcal{O}()$ notation that we saw in class: [10]

Claim

Let $f(n)$, $g(n)$, and $h(n)$ be functions from non-negative integers to real numbers. If $f(n) = \mathcal{O}(g(n))$ and $g(n) = \mathcal{O}(h(n))$, then $f(n) = \mathcal{O}(h(n))$.

5. Prove or disprove:

   (a) $\log_2 3n = \mathcal{O}(\log_2 2n)$ [10]

   (b) $2^{3n} = \mathcal{O}(2^{2n})$ [10]

6. Consider the following problem:

Count Palindromic Substrings

- Input: An integer $n \geq 1$ and a string S of length n. String S is a list indexed from 0; its elements are thus $S[0], S[1], \ldots, S[(n-1)]$.
- Output: The number of different pairs (i, j) ; $0 \leq i < j \leq (n-1)$ such that the substring $S[i]S[i+1]\cdots S[j]$ of S is a palindrome.

   (a) Write the *complete* pseudocode for an algorithm that solves the above problem in $\mathcal{O}(n^c)$ time—in the worst case—for some constant c. You will get the credit for this part only if your algorithm is correct and complete, and runs within the required time bound. [15]

   (b) Prove that your algorithm of part(a) is correct. [15]

   (c) Prove that your algorithm from part (a) runs in $\mathcal{O}(n^c)$ time in the worst case, for a small constant c. For this you may assume that each array operation, and each comparison of a pair of characters, take constant time. Clearly state any other assumptions that you make. [10]