Instructions

This exam has 3 questions for a total of 120 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.
Do not use hash functions or hash tables/maps in your solutions. You may split up the pseudocode for an algorithm into two or more functions, if you wish; you don't have to write the entire pseudocode for an algorithm as one function. You do *not* have to use loop invariants while proving the correctness of algorithms; but you must correctly explain why each loop (if there are some) does what you expect it to do.
Clearly state all assumptions that you make when analysing algorithms.
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.
Warning: CMI's academic policy regarding cheating applies to this exam.

1. Define each of the following terms. You will get the credit for these definitions only if they are clear, correct, and complete. [30]

   (a) Decision problems.
   (b) The class P.
   (c) The class NP.
   (d) The relation $\leq_P$ defined in class.
   (e) The class NP-Hard.
   (f) The class NP-Complete.

2. Consider the following problems:

Vertex Cover

- Input: An undirected graph $G = (V, E)$, and an integer k.
- Question: Does there exist a subset $X \subseteq V$; $|X| \leq k$ such that every edge in E has at least one end-point in X?

Clique

- Input: An undirected graph $G = (V, E)$, and an integer k.
- Question: Does there exist a subset $X \subseteq V$; $|X| \geq k$ such that for every 2-subset of vertices $\{v, w\} \subseteq X$ there is an edge in E with v and w as its end-points?

(a) Assuming that VERTEX COVER is NP-Hard, prove that CLIQUE is NP-Complete using a polynomial-time many-one (Karp) reduction. [10]

(b) Assuming that CLIQUE is NP-Hard, prove that VERTEX COVER is NP-Complete using a polynomial-time many-one (Karp) reduction. [10]

3. The notation $X = X_1 \uplus X_2 \uplus \cdots \uplus X_t$ denotes a partition of set X into t parts. That is, that X is the union of the t *pairwise disjoint*, *non-empty* subsets X_i.

The PARTITIONED PATH problem is defined as follows:

PARTITIONED PATH

- Input: A positive integer k ; $1 \leq k \leq n$, and an undirected graph $G = (V, E)$ on n vertices, given together with a partition of its vertex set as $V = V_1 \uplus V_2 \uplus \cdots \uplus V_k$. Further, every edge in G has its two end-points in *distinct* parts of the partition. In other words: there is no edge in G with both its end-points in any one part of the partition.
- Question: Does there exist a simple path with k vertices, in G? Recall that a simple path is one in which no vertex is repeated.

(a) Write the pseudocode for an algorithm that uses dynamic programming to solve PARTITIONED PATH in worst-case running time $\mathcal{O}(2^k \cdot n^c)$ for some fixed constant c that is independent of n and k. You will get the credit for this part only if your algorithm is correct, uses dynamic programming, and runs within the required time bound. [30]

You may assume that adding an element to a list, and iterating through the elements of a list, each takes time linear in the length of the list. Clearly state any other assumptions that you need.

You may use clear word descriptions of:

- Polynomial-time steps that you wish to perform on the graph and its constituents (vertices, edges, neighbourhoods etc.); you don't have to implement a graph API from scratch!
- Polynomial-time operations that you wish to perform on arrays or lists; you don't have to write out the detailed steps for these.

(b) Argue that your algorithm correctly solves the problem. [20]

(c) Argue that your algorithm runs within the required time bound, in the worst case. Clearly state the assumptions that you need. [20]