Worth: 7.5% **Due:** Before 10pm on **Friday 16 November**.

Remember to write the *full name*, *student number*, and *CDF/UTOR email address* of *each group member* prominently on your submission.
*Please read and understand the policy on Collaboration given on the Course Information Sheet. Then, to protect yourself, list on the front of your submission **every** source of information you used to complete this homework (other than your own lecture and tutorial notes, and materials available directly on the course webpage). For example, indicate clearly the **name** of every student with whom you had discussions (other than group members), the **title** of every additional textbook you consulted, the **source** of every additional web document you used,* etc.

For each question, please write up detailed answers carefully. Make sure that you use notation and terminology correctly, and that you explain and justify what you are doing. Marks **will** be deducted for incorrect or ambiguous use of notation and terminology, and for making incorrect, unjustified, ambiguous, or vague claims in your solutions.

---

[11] 1. For each decision problem D below, state whether $D \in P$ or $D \in NP$, then justify your claim.

- For decision problems in P, describe an algorithm that decides the problem in polytime (including a brief argument that your decider is correct and runs in polytime).
- For decision problems in NP, describe an algorithm that verifies the problem in polytime (including a brief argument that your verifier is correct and runs in polytime), and give a detailed reduction to show that the decision problem is NP-hard — *for your reduction(s), you **must** use one of the problems shown to be NP-hard during lectures or tutorials.*

For each of the following decision problems, recall that a cycle an an undirected graph is *simple* if it contains no repeated vertex or edge.

(a) ExactCycle:

Input: An undirected graph G and a non-negative integer k.

Question: Does G contain some simple cycle on **exactly** k vertices?

(b) SmallCycle:

Input: An undirected graph G and a non-negative integer k.

Question: Does G contain some simple cycle on **at most** k vertices?

(c) LargeCycle:

Input: An undirected graph G and a non-negative integer k.

Question: Does G contain some simple cycle on **at least** k vertices?

[17] 2. Most online shopping sites maintain databases of customers and products purchased. Here is a small example of what such a database might look like: each entry indicates the number of times that a customer purchased a product, within a fixed time period (*e.g.*, during the last 30 days).

	Product 1	Product 2	Product 3	Product 4
Customer A	0	12	1	0
Customer B	3	0	0	1
Customer C	0	0	1	0
Customer D	1	2	4	5

Suppose an advertising company wants to make use of this data to put together a pool of customers for market research: they want to find a group of customers whose interests (as indicated by their recent product purchases) are as diverse as possible.

We call this the "DIVERSITY" problem, and formalize it as follows:

Input: A table of purchases (as above), with $n \geqslant 1$ customers and $m \geqslant 1$ products.

Output: A subset of customers that is as large as possible, such that no two customers in the subset have any purchase in common.

For example, if the input is the table on the previous page, $\{B, C\}$ is a subset of customers with no common purchase that is as large as possible (every subset of three or more customers has some common purchase).

(a) Show how to represent this problem as an Integer Program. Explain how solutions to the original problem and solutions to your integer program correspond to each other.

(b) Give a clear and precise statement for a decision problem that corresponds to the DIVERSITY problem. (HINT: Add a parameter.)

(c) Give a detailed proof that your decision problem is *NP*-complete.

[17] 3. Imagine that we have a network of processors, each one of which needs access to a common database in order to carry out its work (*e.g.*, a network of banking machines accessing client information). To simplify, assume all operations are queries (no operation changes the data). Our problem is to determine which processor(s) should store the database (these processors will be called "servers") in order to have reasonably fast access time (including the load on each server), as well as having as little duplication of the database as possible.

At one extreme, each processor could store its own copy of the database. This would ensure the fastest possible access time, but at the cost of replicating the database many times. At the other extreme, the database could be stored in a single server (all other processors would connect to the server to access the data). This would ensure the smallest amount of duplication, but at the cost of longer access times and server load. In either case, there is a significant cost if the network is large.

A reasonable middle ground between the two extremes would be to fix a distance parameter d and to store the database on as few servers as possible, chosen so that every processor is within distance d of at least one server (the distance is measured as the minimum number of links required to send communications from one processor to the next). Unfortunately, it is impossible to solve this problem efficiently.

Formally, show that the following "SERVERLOCATION" problem is *NP*-complete — write a detailed proof.

Input: An undirected graph G (the network of processors) and non-negative integers k and d.

Question: Is there some subset S of no more than k vertices (the "servers") such that every processor is within distance d of some server in S? (The "distance" between vertices u, v is measured as the number of edges on a shortest path from u to v, or infinity (∞) if there is no path from u to v.)