This is a closed book exam: the only resources allowed are blank paper, pens, and your head, but you may use a handwritten A4 page with information that you consider useful for solving the exam exercises. Explain your reasoning. Write clearly, in the sense of logic, language, and legibility. The clarity of your explanations affects your grade. Good luck!

Write your name and student number on *all* solution sheets and here. Name: .....
At the end of the exam, hand in *all* sheets that you received, including this one. Student number: .....

Problem 1 [30%] Design the Entity-Relationship schema of an application for managing mountain excursions, for which the following information is of interest. Of each *excursion*, we are interested in the code (identifier), the cost, the persons who took part in the excursion (at least one), and the mountain climbings it includes (at least one). For each excursion, a maximum of one climbing per day takes place, and of each such *climbing* (which is specific to that excursion) we are interested in the date on which it took place, the duration, and the main mountain climbed. In addition, there are *special climbings*, for which we are interested also in the required skill level and the mountains (at least one) climbed in addition to the main one (these mountains are called secondary mountains of the climbing), each with the extra climbing time and the person who acted as guide in the climbing of that secondary mountain. Note that a person may not act as guide in more than one climbing of a secondary mountain on the same day (but they may on different days). Of each *mountain*, we are interested in the name (identifier), the height, and the GPS coordinates. Of each *person*, we are interested in the ssn (identifier), the first name, the surname, and the age.

Problem 2 [40%] Carry out the logical design of the database, producing the complete relational schema with constraints, taking into account the following indications: *(i)* every time we access a climbing, we are also interested in its main mountain; *(ii)* every time we access the information about a climbing, we always want to know whether it is a special climbing, and if so, we want to know the required skill level.

In your design you should follow the methodology adopted in the course, and you should produce:

1. [7%] the restructured Entity-Relationship schema (possibly with external constraints),
2. [25%] the direct translation into the relational model (possibly with external constraints), and
3. [8%] the restructured relational schema (again with constraints).

You should motivate explicitly how the above indications affect your design.

Problem 3 [20%] Consider a database B containing the two tables: *(i)* `Nodes(node)`, which stores all the nodes of a directed graph G, and *(ii)* `Edges(start,end)`, which stores all the edges of G, where an edge from node n_1 to node n_2 is represented by the tuple $t = \langle n_1, n_2 \rangle$ in the relation `Edges` for which $t.\texttt{start} = n_1$ and $t.\texttt{end} = n_2$.

If G contains the edge from n_1 to n_2, then n_2 is said to be a *successor* of n_1 in G and n_1 is said to be a *predecessor* of n_2 in G. Furthermore, a node is said to be a *source* if it has at least one successor and no predecessor.

We know that the database B satisfies both

(i) the foreign key constraint from `start` of `Edges` to `node` of `Nodes`, i.e., $\texttt{Edges}[\texttt{start}] \subseteq \texttt{Nodes}[\texttt{node}]$, and
(i) the foreign key constraint from `end` of `Edges` to `node` of `Nodes`, i.e., $\texttt{Edges}[\texttt{end}] \subseteq \texttt{Nodes}[\texttt{node}]$.

Answer the following questions.

1. [8%] Write a *SQL* query that returns the average number of predecessors of the nodes of G (remember to consider also nodes having no predecessors).
2. [12%] Write a *relational algebra* query that computes all nodes of G that have as predecessors only source nodes (i.e., all nodes of G that have no predecessor that is not a source node).

Problem 4 [10%] Still referring to the database B described in Problem 3, suppose the foreign key constraint from `start` of `Edges` to `node` of `Nodes` is defined as `ON DELETE RESTRICT`, while the foreign key constraint from `end` of `Edges` to `node` of `Nodes` is defined as `ON DELETE CASCADE`.

Answer the following questions.

1. [5%] Is it always possible to obtain the deletion of any given tuple from the `Nodes` relation of graph G by executing only operations of the form "`DELETE FROM Nodes WHERE ...`"? If the answer is positive, give reasons; if it is negative, say which nodes can be deleted and which cannot, again giving reasons.

2. [5%] Answer the question in the previous point when the graph G is a tree, which, remember, is a particular graph in which an edge from n_1 to n_2 indicates the hierarchical relationship between the parent node n_1 and the child node n_2.