Further details of *Some* Projects:

***1.1 Set Partitioning Structure*:**

(A) Description

A data structure to capture how various sets of objects (elements) could be [mathematically] partitioned into disjoint subsets. Sets and the partitions themselves can also be also elements of other sets. Each element has a unique ID (UID), attributes unique to it, and lists of sets and of partitions to which it is associated. Each collection (or partition) has its list of element, and attributes pertaining to the collection (or partition) as a whole (e.g., average height). The elements of a partition is a list of [sub]sets—the partitions—that make it up, and from them the members of each partition. (Sets are of three categories: class, sets?, instances)

(B) A data structure that has the following components:

(1) An item, which is one of the following:

1) [Basic] item: The descriptor of a basic item, the Basic Item Descriptor, contains:

- A unique integer ID.
- An abstract description of what the item is. (e.g., as a terse string.)
- Master set to which item belongs on creation.
-
- List of IDs of set [items] to which the basic item belongs.
- Attributes/properties of item. (e.g., if a person, age, sex, height), possibly in a variant record.

2) Set [item]: A set item could be treated as a basic item, and so has all the attributes of a basic item. It could also be treated as a set, and so has attributes that describe structures and elements within the set. Thus, the descriptor of a set item, the Set Item Descriptor, contains the following:

- Basic Item Descriptor (for the set item). This descriptor is as described in (1) above.
- Set Component Descriptor. This descriptor holds information on the structure and elements of the set. It contains:
  - A collection (i.e., set or list) of IDs of elements in the set.
  - A collection (i.e., set or list) of IDs of partitions (i.e., partition items) the set has.
  - A collection (i.e., set or list) of attributes common to elements of the set (e.g., if persons, average height, age bracket, gender.) and applicable to all elements of the set.
- An abstract description of what the elements of the set is. (e.g., as a terse string.)
-

3) Partition [item]: A partition item could be treated as a basic item, and so has all the attributes of a basic item. It could also be treated as a [mathematical] partition—a relation where all elements of a given set belong to exactly one of its subsets, and the union of all its subsets is the set. Thus, the descriptor of a partition item, the Partition Item Descriptor, contains the following:

- Basic Item Descriptor (for the partition item). This descriptor is as described in (1) above.
- Set Relation Descriptor. This descriptor holds information on the structure and elements of the set. It contains:
  - A collection (i.e., set or list) of IDs of sets (as in (2)) that this relation partitions.
  - For each of the sets partitioned, a corresponding collection (i.e., set or list) of IDs of sets that are subsets of the partition.
  - A collection (i.e., set or list) of attributes that describes the partitioning (e.g., if persons,
- A formal description of the relation (e.g., set comprehension relation; generator function of partition subsets, etc.

(C) Basic Operations

We expect students to develop operations (procedures and functions) that can be used to access the

data structure. However, we do not expect these operations to be comprehensive.
(1) Basic Items:
- Create/destroy basic items;
- Define operations to access and/or modify entries of basic item.

(2) Set Items:
- Create/destroy set items; define operations to access and/or modify entries of set items.
- Add/remove element in a set; move an element from one set to another.
- Locate an element as a member of a set (or partition).
- Other operations the student may care to include.

Note that updating data in a given set might entail updating (i.e., add/remove/modify) entities associated somehow with the set (e.g., elements, partitions). For example, adding an element u to a given set X also entails adding the ID of X to the list of IDs of the set items to which u belongs.

(3) Partition Items:
- Create/destroy partition items; define operations to access and/or modify entries of partition items.
- Add/remove element in a partition; move an element from one partition to another
- Modify the definition of a partition or the set over which it is applied.
- Locate an element as a member of a set (or partition).
- Other operations the students may care to include.

2.1 Execution tunnelling.

(A) Description

Inspired by the OSI ISO 7 layer networking model where specific data communication services are provided at specific levels and services needed from other levels are invoked (called) from such levels, we want to generalise the invocation of services to other scenarios (e.g., security clearances/ access control) where the conditions and where [in the layered model] services can be called are constrained. The idea is to "tunnel" through service layers [and their constraints] to a desired layer and invoke a service of that layer, while making sure any constraints would have been respected or duly relaxed.

An example illustrates the point, and also help define our assumptions. "Bidirectional" services at a given layer can only *directly* invoke services from adjacent layers. Thus, bidirectional service bs3 in layer 3 can call services from layers 2 and 4, which are in adjacent layers, but cannot invoke services from layers 1, 5 or 6. For "hierarchical" services, only services from adjacent higher layers [in the hierarchy] can call lower ones. Assume layers are numbered in their increasing hierarchical order. Then, hierarchical service hs3 would be able to call service s2 from layer 2, but not services s3 from layer 3 (hierarchically superior) or service s1 from layer 1 (not adjacent).

We consider bidirectional services, but can easily specialise to specific hierarchical ones too. For this project exercise, we are happy to use, *initially*, procedure/function stubs (i.e., with dummy content) to stand for services, and basic integer values to stand for arguments and context information (constraints, etc.) passed to services.

To enable the execution of service s2 of service layer 2 from service layer 4, say, "tunnel [down]" its execution through service layer 3 to service layer 2. To do this, we create dummy services s2_4 and s2_3 at layers 4 and 3 respectively. (We can also "tunnel up" execution to layer 6.) The [dummy] service at each layer expects an extra argument to handle context issues at each layer.

More generally, we automatically append "_n" to service name "x" to get a dummy service name "x_n" of layer n. (Of course, there are more elegant ways of doing this, with function variables