B561 Assignment 7
Relational Programming

1. Consider the relation schema `Graph`(source INTEGER, target INTEGER) representing a directed graph. The graph *Graph* is *connected* if for each pair of nodes (s, t) in *Graph*, there exists a path in *Graph* from s to t.

   An *articulation point* of *Graph* is a node n in *Graph* such that removing the edges in the graph with source n as well are removing the edges with target n results in a graph that is **not** connected. Write a Postgres program that determines the articulation points of *Graph*.

2. Consider the following relational schema. A tuple $(pid, cpid)$ is in `Parent_Child` if *pid* is a parent of child *cid*.

   ```
   Parent_Child
   -----------
   |PId | SId |
   ------------
   ```

   You can assume that the domain of PId and SId is INTEGER.

   Write a Postgres program that computes the pairs (id_1, id_2) such that id_1 and id_2 belong to the same generation in the `Parent-Child` relation and $id_1 \neq id_2$. (id_1 and id_2 belong to the same generation if their distance to the root in the `Parent-Child` relation is the same.)

3. Consider a unary relation $A(x)$ of integers.

   Using arrays to represent sets, write a program `powerset` in Postgres to compute the powerset of A.

   For example, if A is the following relation

   A

x
1
2
3

   then the output should be the following complex-object relation PowersetA(subset integer[])

PowersetA
subset
$\{\}$
$\{1\}$
$\{2\}$
$\{3\}$
$\{1,2\}$
$\{1,3\}$
$\{2,3\}$
$\{1,2,3\}$

4. Suppose you have a weighted undirected graph $Graph = (V, E)$ stored in a ternary table named `Graph` in your database. A triple (n, m, w) in Graph indicates that $Graph$ has an edge (n, m) where n is the source, m is the target, and w is the edge's weight. (In this problem, we will assume that each edge-weight is a positive integer.) Since the graph is undirected, whenever there is an edge (n, m, w) in the graph, then (m, n, w) is also in the graph. Below is an example of a graph $Graph$.

`Graph`

Source	Target	Weight
0	1	2
1	0	2
0	4	10
4	0	10
1	3	3
3	1	3
1	4	7
4	1	7
2	3	4
3	2	4
3	4	5
4	3	5
4	2	6
2	4	6

A *spanning tree* T of $Graph$ is a sub-graph of $Graph$ that is acyclic and such that for each node n in $Graph$ there is an edge in T of the form (n, m) or (m, n). I.e., each node of $Graph$ is the end point of an edge in $Graph$. The weight of a sub-graph of $Graph$ is the sum of the weights of the edges of that sub-graph. A *minimum spanning tree* of $Graph$ is a *spanning tree* of $Graph$ of minimum cost.

Write a Postgres program that determines a minimum spanning tree of a graph $Graph$. You can use Prim's Algorithm to determine a spanning tree. Consult https://en.wikipedia.org/wiki/Minimum_spanning_tree

and

https://en.wikipedia.org/wiki/Prim's_algorithm.

5. Consider the heap data structure. For a description, consult

   https://en.wikipedia.org/wiki/Binary_heap.

   There are many other sites (including video) that discuss how this data structure works.

   (a) Implement this data structure. This implies that you need to implement the `insert` and `extract` operations.

       You are **not** allowed to use arrays to implement this data structure!

   (b) Implement the heap sort algorithm.

       You are **not** allowed to use arrays to implement this data structure!

   The input format is a list of integers stored in a binary relation Data(index,value).

   For example Data could contain the following data.

Data

index	value
1	3
2	1
3	2
4	0
5	7

   The output of the sort function should be stored in a relation SortedData(index,value). On the Data relation above, the sort function should return the relation.

SortedData

index	value
1	0
2	1
3	2
4	3
5	7