**Project 03: Role Based Access Control (RBAC)**

**Introduction**

We set the roles and permissions for all the users that participate in a conference (i.e., chairs, authors, etc.). We consider the following sets:

* the set of users USERS
* the set of roles ROLES
* the set of permissions PERMISSIONS
* the set of user-role pairs UR is subset of USERS \* ROLES
* the set of role-permission pairs RP is subset of ROLES \* PERMS
* a set of pairs of roles RH (role hierarchy), called ascendant and descendant roles, where an ascendant role inherits permissions from a descendant role. This role hierarchy RH is transitive and may contain cycles.

For any user the set of roles of that user and their transitive descendant roles is called the AuthorizedRoles for that user. After we decided on the above sets and assigned the roles to all the users, we realized that we defined too many roles for each user and too many permissions for every role because some users are involved in multiple roles (author, chairs, reviewers, etc.), so we want to simplify the access to permissions such that each user has the same permissions through AuthorizedRoles as before. Create a single layer of roles and attach to each user a single role. Some users have the same permissions, so use a single role for those users.

**Input format**  
  
An input file for Prolog contains the following facts:

* one fact of the form users(Number) containing the number of users (these users are represented by numbers from 1 to Number).
* one fact of the form roles(Number) containing the number of roles (these are represented by numbers from 1 to Number).
* one fact of the form perms(Number) containing the number of different permissions (these are represented by numbers from 1 to Number).
* facts of the form ur(User,Role) containing the user-role pairs.
* facts of the form rp(Role,Permission) containing the role-permission pairs.
* facts of the form rh(Role1,Role2) containing the ascendant and descendant roles.

**Output format**

* A predicate of the form authorized\_roles(User,List\_Roles) that returns the set of roles for the given user. **(30 points)**
* A predicate of the form authorized\_permissions(User,List\_Permissions) that returns the set of permissions for the given user. **(30 points)**
* A predicate minRoles(S) where S is the smallest number of roles needed to cover all users. **(30 points)**

**Report (5 points)**The report should have the analysis for each output predicate. Use the test file (test.pl) you will submit for your analysis, giving details about how the output predicates result in the right answers within a reasonable time (few seconds).

**Documentation (5 points)**Your code will be reviewed for comments and good documentation.