Homework VIII
100 points

For this project you are to write a Prolog program that determines a spanning tree for a given graph. A graph $G=(V, E)$ is a pair consisting of a finite set $V$ of vertices or nodes, plus a finite set $E$ of edges, each edge joins two nodes of the graph and is unoriented (may be traversed in either direction). A path from one node to another is a sequence of adjoining edges that start at the first and end at the second — in the graph below a path from $a$ to $f$ is provide by the edges $(a,b)$, $(b,c)$, $(c,d)$, $(d,f)$, and this is in fact a simple path. A simple path is one in which no edge is repeated. A cycle is a path which starts and ends at the same node. Normally we understand that a null path is the sequence with no edges and trivially joins each node to itself. A graph is connected if there is a path between each pair of distinct nodes. A spanning tree of graph $G=(V, E)$ is a subgraph $G'=(V, E')$, where $E'\subseteq E$, which is a tree. Each connected graph will have a (commonly several) spanning tree(s).

For this project, a graph $G=(V_S, E)$ will be defined by two predicates, vertices($V_S$), where $V_S$ is a list of all the vertices, and predicate edge($v_1, v_2$) which is true for each $(v_1,v_2)\in E$. The predicate spanTree($T_S$) should instantiate $T_S$ to a list of edges which constitute a spanning tree, or fail if there is none. For the graph depicted above, the query ?- spanTree(T_S). should yield 'yes', and might bind $T_S=\{(a,b), (b,f), (c,d), (c,f), (e,f)\}$, the spanning tree shown in bold in the figure above.

```
Graph depiction

Graph representation

vertices([a,b,c,d,e,f]).

edge(a,b).
edge(a,f).
edge(b,c).
edge(b,f).
edge(c,d).
edge(c,f).
edge(d,e).
edge(d,f).
edge(e,f).
```
The following four properties are all logically equivalent characterizations of trees:

- a tree is a connected graph with no non-null simple cycles
- a tree is a graph with exactly one simple path between each pair of distinct nodes
- a tree is a connected graph with n nodes and n-1 edges
- a tree is a graph where the addition of any edge creates a cycle, and the deletion of any edge disconnects the graph

Roughly speaking, these conditions may be interpreted as saying that a tree accomplishes connectivity with as few edges as possible. This suggests two alternative ways of approaching a solution to finding a spanning tree:

1. start with no edges, and selectively add edges of G until connectivity is achieved, but without creating cycles
2. start with all edges of G and selectively delete edges until there are no cycles, but always maintaining connectivity

You should write and test a Prolog program for finding a spanning tree of any given graph – if course, it may (must) fail if the graph is not connected. The spanning tree should be presented as a list of edges selected from the graph that form a tree on its vertices.

Program submission instructions
Program solutions must include documentation that makes it clear both what general method you used in constructing the program, and how the details of the program accomplish that method. You need to run test cases that exercise every component of your code, and include documentation that justifies that your test data meets this condition (the UNIX 'script' command is normally used to prepare materials). It is not the grader’s responsibility to figure out how you wrote the program and whether it is correct — it is your responsibility to explain your program and convince the grader it is completely tested and correct. Full credit will not be awarded, even for (apparently) correct programs, without completing these requirements.

To receive a score you must turn-in a listing of your source program and test outcomes (paper copy). In addition, you should use the 'submit' command to provide an electronic copy of your source code. Send it to the directory Hwk8 for course c054. Your identification and the time of submission are automatically attached to your electronic submission. There is a link to the description of the submit command on our class Web page.