/home/shawn/schoolwork/CS4216/PS4/beep.c

```c
#include <stdio.h>
int main() {
	printf("%c",'\a');
}

```

/home/shawn/schoolwork/CS4216/PS4/ex2.pl

```
/*
 * Collaborative work:
 *      Benjamin Tan
 *      Shawn Tan
 *
 */

:-lib(ic).
:-lib(branch_and_bound).
constraints(L,M,Tree,NewTree) :-
   traverse(0,L,Tree,NewTree,VarList),
   x_constraints(VarList,M,Width),
   term_variables(VarList,Vars),
   length(Vars,Len),
   MaxWidth is Len * M,
   Width :: 0..MaxWidth,
   ic:max(Vars,Width),
   write(NewTree),nl,
   write(Vars),nl,
   minimize(search(Vars,0,first_fail,indomain,complete,[]),Width).


perm([],[]).
perm(L,[H|T]) :- delete(H,L,R),perm(R,T).
search(V) :- perm(V,VP), ( foreach(X,VP),param(VP) do get_min(X,X) ).

x_constraints(XList,M,Width) :-
   (
      foreach([H|T],XList),param(M,Width) do
      H #>=0,
      (
         fromto(
            (H,T),
            (P,[C|Rest]),
            (C,Rest),
            (Last,[])
         ),param(M) do
         C #>= P + M
      ),
      Last #=< Width
   ).

traverse(Depth,L,Leaf,NewLeaf,[[X]]) :- atom(Leaf),Y is Depth*L, NewLeaf =.. [Leaf,X,Y].
traverse(Depth,L,Tree,NewTree,XList) :-
   Tree =.. [Node|Children],
   Depth1 is Depth+1,
   Y is Depth*L,
```

```
   (
      fromto(
         (Children,[],Args),
         ([C|T],VarIn,[Arg|ArgOut]),
         (T,VarOut,ArgOut),
         ([],VarList,[])
      ),param(Depth1,L) do
      traverse(Depth1,L,C,Arg,Vs),
      combine_list(VarIn,Vs,VarOut)
   ),
   append(Args,[X,Y],NewArgs),
   align_center(X,VarList),
   NewTree =.. [Node|NewArgs],
   XList = [[X]|VarList].

align_center(X,Desc):-
   Desc = [Children|_],
   Children = [First|_],
   append(_,[Last],Children),
   First #=< X,
   Last  #>= X,
   (First == Last ->
      true;
      X #= (First + Last)/2
   ).

combine_list(Lists1,Lists2,CombList) :-
   (
      fromto(
         (Lists1,Lists2,CombList),
         (In1,In2,[Comb|OutRes]),
         (Out1,Out2,OutRes),
         ([],[],[])
      ) do
      (In1 = [] -> I1 = [],Out1=[];In1 = [I1|Out1]),
      (In2 = [] -> I2 = [],Out2=[];In2 = [I2|Out2]),
      append(I1,I2,Comb)
   ).
```