

Worth: 10% (together with part I of this assignment)

Due: By 1pm on Wednesday **25 March**

For the following questions, you must use good Prolog style in your code (in particular, make appropriate use of unification) including good external and internal comments (to explain the purpose of your code and your decisions on how to implement it), as well as good visual layout (formatting, indenting, blank lines) to make your code easy to read and understand.

1. Write a predicate `last_member(X,L)` that is true iff `X` is the last element in list `L`.
Submit your code for this question in file “`last_member.pl`”.
2. Write a predicate `trib(N,X)` that is true iff `X` is the `N`-th “standard” Tribonacci number (with base cases $T_{-1} = -1, T_0 = 0, T_1 = 1$), for any value of `N` (negative, zero, or positive)—note that Prolog automatically performs computations on arbitrary-sized integers when necessary. For full marks, your computation must take at most **linear time** as a function of `N`. (HINT: It is fine to define an auxiliary predicate to perform the bulk of the computation.)
Submit your code for this question in file “`trib.pl`”.
3. Write a predicate `sum_list(List,Total)` that is true iff `Total` is the sum of every integer value stored in `List`. (HINT: In Prolog, `integer(X)` is true iff `X` is instantiated to an integer value.)
Submit your code for this question in file “`sum_list.pl`”.
4. Consider again our family relationship example, slightly expanded.

```

male(albert).
male(bob).
male(charlie).
male(daniel).
male(steven).

female(alice).
female(bianca).
female(brigit).
female(carol).
female(jane).
female(suzy).

parent(albert,bob). % albert is a parent of bob...
parent(alice,bob). % ...and so on...
parent(albert,brigit).
parent(alice,brigit).
parent(carol,daniel).
parent(charlie,daniel).
parent(bianca,jane).
parent(bob,jane).
parent(brigit,steven).
parent(daniel,steven).
parent(brigit,suzy).
parent(daniel,suzy).

```

Write code for the following predicates:

```

mother(M,Y) :- % M is the mother of Y

father(F,Y) :- % F is the father of Y

child(C,Y) :- % C is a child of Y

son(S,Y) :- % S is a son of Y

daughter(D,Y) :- % D is a daughter of Y

sibling(X,Y) :- % X is a sibling of Y

brother(B,Y) :- % B is a brother of Y

sister(S,Y) :- % S is a sister of Y

uncle(U,Y) :- % U is an uncle of Y

aunt(A,Y) :- % A is an aunt of Y

nephew(N,Y) :- % N is a nephew of Y

niece(N,Y) :- % N is a niece of Y

grandparent(G,Y) :- % G is a grandparent of Y

grandchild(G,Y) :- % G is a grandchild of Y

cousin(C,Y) :- % C is a cousin of Y

```

For `sibling/2`, make sure that the goal `sibling(X,X)` does *not* succeed, and that for all siblings `X,Y`, the goal `sibling(X,Y)` will succeed only once. (HINT: In Prolog, the goal `X == Y` succeeds iff `X` and `Y` are fully instantiated and evaluate to the same value. The goal `X \== Y` succeeds iff the goal `X == Y` fails. To simplify this question, you may assume that if `Z`'s parents are `X` and `Y`, then every sibling of `Z` also has parents `X` and `Y`, *i.e.*, do not try to account for the possibility of step-siblings.)

Make sure that the same holds for `cousin/2`, *i.e.*, `cousin(X,X)` should fail, and for all cousins `X,Y`, `cousin(X,Y)` should succeed only once.

Submit your code for this question in file "family.pl" (include the facts given at the start of this question in your file).

5. Draw a Prolog search tree for the query `?- cousin(C,suzy).`, for the facts and predicates in the previous question.

Submit your search tree on paper, at *beginning* of the lecture on the due date.