

EECS-3401A
Introduction to AI & LP
York University
Fall 2022

Assignment #2
Widgets

General

For **Q1** through **Q7**, build **Prolog** procedures for **SWI Prolog** that work as specified.

Part I [20pt]: Dr. Dogfurry's Binary Tree

Dr. Dogfurry has designed a representation for binary trees. A “node” term looks like

- `[bt(integer, binary-tree, ...)]`
- `[]`

The latter represents an empty binary tree. The former represents a *binary-tree* node. Call

A binary-tree node can be represented as `[bt(Key, Left, Right)]`, where `Key` is an integer, `Left` is a *right-child* binary tree (the third argument is placed there.)

E.g., `[bt(5, [], [bt(10, [], [])])]`

For the following, you will be given a binary tree, apropos Dr. Dogfurry's representation.

- it is *finite* (that is, it has a finite number of nodes)
- if it is not the empty tree, its first argument is a proper integer and its second and third arguments are proper binary trees
- that it is *ground*.



... A “node” term

... arguments are

... a *right-child* binary tree, `[]`,

... a proper binary

... the next two

Q1. [5pt] empty_bintree/1

Write a procedure for `empty_bintree/1` that takes an argument `Tree`. It should return *true* if this is an *empty* binary tree, apropos Dr. Dogfurry's representation; and *false* (*fail*), otherwise.

The procedure should return *true* — with the appropriate additional behaviour — with an *unbound* variable for its argument.

Q2. [5pt] bintree_contains/2

Write a procedure for `bintree_contains/2` that takes two arguments, `Key` and `Tree`, in that order. The latter argument is *input*.

The procedure should evaluate *true* iff `Key` is equal to the *key* of any node in the *binary tree*