1 The Module `kindergarten_garden.pl`

`plant(+Plant:atom, -Encoding:char)` kindergarten_garden.pl

Four different types of seeds are planted:

Plant	Prolog encoding	Diagram encoding
Grass	`grass`	G
Clover	`clover`	C
Radish	`radishes`	R
Violets	`violet`	V

```
:- module(kindergarten_garden, [garden/3]).
%! plant(+Plant:atom, -Encoding:char) is det.
%
% Encode a plant name as a single character.
plant(grass, 'G').
plant(clover, 'C').
plant(radishes, 'R').
plant(violets, 'V').
```

```
find_child_plants(+Child:atom,
          +FirstRow:list(char),
          +SecondRow:list(char),
          +Children:list(atom),
          -Plants:list(atom))
```
kindergarten_garden.pl

Find `Child`'s `Plants`, given two rows of plants and a list of `Children`.

If `Child` is the first of `Children`, their `Plants` are the first two in each row.

```
%! find_child_plants(+Child:atom, +FirstRow:list(char), +SecondRow:list(char),
%!     +Children:list(atom), -Plants:list(atom)) is det.
%
% Find =Child='s =Plants=, given two rows of plants and a list of =Children=.
find_child_plants(Child, [P1, P2  |_], [P3, P4  |_],
                  [Child  |_], [P1, P2, P3, P4]) :-
    !.
```

Otherwise, recursively check the next set of plants.

```
find_child_plants(Child, [_, _  |FirstRow], [_, _  |SecondRow],
                  [_  |Children], Plants) :-
    find_child_plants(Child, FirstRow, SecondRow, Children, Plants).
```

```
garden(+Garden:string, +Child:atom,
       -Plants:list(atom))
```
kindergarten_garden.pl

Determine which `Plants` in the `Garden` belong to the given `Child`.

```
%! garden(+Garden:string, +Child:atom, -Plants:list(atom)) is det.
%
```

```
% Determine which =Plants= in the =Garden= belong to the given =Child=.
garden(Garden, Child, Plants) :-
```

List all the `Children`.

```
    Children = [alice, bob, charlie, david, eve, fred,
                ginny, harriet, ileana, joseph, kincaid, larry],
```

Split the `Garden` into its two lines.

```
    split_string(Garden, "\n", "", [FirstLine, SecondLine]),
```

Convert both lines into rows of encoded plants.

```
    string_chars(FirstLine, FirstRow),
    string_chars(SecondLine, SecondRow),
```

Determine which `EncodedPlants` belong to the given `Child`.

```
    find_child_plants(Child, FirstRow, SecondRow, Children, EncodedPlants),
```

Convert the list of `EncodedPlants` into a list `Plants` of plant names.

```
    maplist(plant, Plants, EncodedPlants).
```