Lab 7 – 08226 Artificial Intelligence

This lab tutorial will introduce you to using lists in your Prolog programs.

1.0 List Basics

Start SWI-Prolog-Editor from the Windows Menu system and create a new Prolog file called **Lab7.pl** and store it in **G:/08226/Lab 7/**.

In Prolog, a list is an ordered collection of elements of any length. Furthermore, elements can be of any mixed type, including other lists, within the list. Lists are often used as array, since Prolog does not have any form of array structure.

A list is represented by square brackets with comers separating the elements within the list. The following are examples of lists with correct syntax:

```
[1,2,3,4,5]

[one,2,three,4,five]

[one,2,three,[4,Variable]]
```

2.0 List Handling

In Prolog, lists are handled by splitting the head of the list from the tail of the list. This is denoted by the vertical line **|**.

Consider the following:

```
[Head | Tail].
```

This list would have the head of the list held in **Head** as a list, and the tail (or the rest of the list) would be held in **Tail** as a list. The following are examples of correct syntax:

```
[1 | [2,3,4,5]]

[one | [2,three,4,Variable]]
```

What these are actually doing is adding the head to the tail into a new list.

Enter the following query:

```
?- List = [1 | [2,3,4,5]].
```

Prolog will return the result **List = [1, 2, 3, 4, 5].** as we would expect as the head (1) has been added to the tail (2,3,4,5) into the new list held in the **List** variable.

2.1 List Exercise 1

Write down the full enumerated lists of the following (e.g [1,2,3,4,5]):

1) `darren mike dawn`
2) `g k w t y`
3) `55 23 0 46 seven`

Write down the following in the [Head | Tail] form:

4) `darren mike dawn`
5) `g k w t y`
6) `55 23 0 46 seven`

Are the following correct list syntax?

7) `[n,m,o]`	**yes / no**
8) `[n	[m,o]]`	**yes / no**
9) `n`	**yes / no**
10) `[f	k,l]`	**yes / no**
11) `[y]`	**yes / no**
12) `[Y]`	**yes / no**
13) `[darren	mckie]`	**yes / no**

Check your answer here: **List Exercise 1 Answers**

3.0 Writing Out the Contents of a List

In Prolog, most list operations and done using recursion, and writing out the contents of a list is no exception. A list is an ordered structure so we cannot access an element in the list directly other than the head of the list.

Consider the following:

```
writelist([]).

writelist([Head|Rest]) :- write(Head), nl, writelist(Rest).
```

This will vertically write a list out to the query window. The program starts with the fact **writelist([])**. The **[]** denotes an empty list in Prolog. Therefore, if we have an empty list this fact will return true. The program finishes with a rule that splits the given list into the head (**Head**) and the rest (**Rest**) of the list, then writes out the head followed by a new line and then recursively calls itself with the remainder of the list.

Enter the following query:

```
?- writelist([1,2,3]).
```

This will vertically display **1** then **2** then **3**. Let's see what is going on. Firstly we have the following:

```
writelist([1 | [2,3]]) :- write(1), nl, writelist([2,3]).
```

This results in **1** being displayed in the query window followed by a new line, and then with a recursive call to itself with the rest of the list namely **[2,3]**. This results in the following:

```
writelist([2 | [3]]) :- write(2), nl, writelist([3]).
```

This results in **2** being displayed in the query window followed by a new line, and then with a recursive call to itself with the rest of the list namely **[3]**. This results in the following:

```
writelist([3 | []]) :- write(3), nl, writelist([]).
```

This results in **3** being displayed in the query window followed by a new line, and then with a recursive call to itself with the rest of the list namely **[]**. This results in the following:

```
writelist([]).
```

This fact returns true and all of the recursive calls will unwind themselves.

To understand the above description, **Trace** the above example of:

```
?- writelist([1,2,3]).
```

3.1 List Exercise 2

Write down a Prolog program that displays the contents of a list in a horizontal manner with two white spaces in between each element of the list. Therefore if we give the program the list [5,6,7,8] it will result in the following output:

```
5  6  7  8
```

Check your answer here: **List Exercise 2 Answers**

4.0 Reading In the Contents of a List from the User

In Prolog, it is often required to read in the contents of a list from the user, one element at a time.

4.1 List Exercise 3

Write down a recursive Prolog program that reads in the contents of a list one element at a time. The program should have a fact in the form **getlist([])** and a rule in the form **getlist([Head|Rest])**. The rule should use the predicates **read()** and **not()**. The program should have a starting rule in the form:

```
go :- getlist(List),

      writelist(List).
```

To enter and display a list the user should enter the following into the query window:

```
?- go.
```

Check your answer here: **List Exercise 3 Answers**

5.0 List Exercise 1 Answers

Write down the full enumerated lists of the following (e.g [1,2,3,4,5]):

1) `darren mike dawn`	**`[darren,mike,dawn]`**
2) `g k w t y`	**`[g,k,w,t,y]`**
3) `55 23 0 46 seven`	**`[55,23,0,46,seven]`**

Write down the following in the [Head | Tail] form:

4) `darren mike dawn`	**`[darren	[mike,dawn]]`**
5) `g k w t y`	**`[g	[k,w,t,y]`**
6) `55 23 0 46 seven`	**`[55	[23,0,46,seven]`**

Are the following correct list syntax?

7) `[n,m,o]`	**yes**
8) `[n	[m,o]]`	**yes**
9) `n`	**no** – has to be enclosed by []
10) `[f	k,l]`	**no** – the tail has to be enclosed by []
11) `[y]`	**yes**
12) `[Y]`	**yes**
13) `[darren	mckie]`	**no** – the tail has to be enclosed by []

Return to your exercise here: **List Exercise 1**

6.0 List Exercise 2 Answers

Write down a Prolog program that displays the contents of a list in a horizontal manner with two white spaces in between each element of the list. Therefore if we give the program the list [5,6,7,8] it will result in the following output:

```
5  6  7  8
```

```
writelist([]).

writelist([Head | Rest]):- write(Head), write('  '), writelist(Rest).
```

Return to your exercise here: **List Exercise 2**

7.0 List Exercise 3 Answers

Write down a recursive Prolog program that reads in the contents of a list one element at a time. The program should have a fact in the form **getlist([])** and a rule in the form **getlist([Head|Rest])**. The rule should use the predicates **read()** and **not()**. The program should have a starting rule in the form:

```
go :- getlist(List),
      writelist(List).
```

To enter and display a list the user should enter the following into the query window:

```
?- go.
```

Answer:

```
go :- getlist(List),
      writelist(List).


getlist([Head|Rest]) :- read(Item), not(Item = end),
                        Head = Item, getlist(Rest).
getlist([]).  % This is placed after the above rule because
              % otherwise the above rule would never be called


writelist([]).
writelist([Head|Rest]) :- write(Head), write('  '), writelist(Rest).
```

Return to your exercise here: **List Exercise 3**