Assignment 3

Brian Moyles – 21333461

Question 1

1. **Write a prolog rule called 'teaches' that returns true if a given instructor teaches a given student**

teaches(X,Y) :- instructs(X,Course), takes(Y,Course).

```
?-
|    teaches(bob,tom).
true
```

2. **Write a prolog query that uses the 'teaches' rule to show all students instructed by bob.**

teaches(bob,Student).

```
?- teaches(bob,Student).
Student = tom ;
Student = mary ;
Student = joe.
```

3. **Write a prolog query that uses the 'teaches' rule to show all instructors that instruct mary.**

teaches(Y,mary).

```
?- teaches(Y,mary).
Y = bob ;
Y = ann.
```

4. **What is the result of the query: teaches(ann, joe).**
   Why is this the case?

The result is false

```
?- teaches(ann,joe).
false.
```

This is because ann instructs ct345, while joe does not take ct345.

The courses joe takes is compared to the courses ann teaches. The query returns false due to the courses not being the same for the instructor and student, in this case ann and joe.

5. **Write a prolog rule called 'classmates' that returns true if two students take the same course. Demonstrate with suitable queries that this rule works as described.**

classmates(X,Y) :- takes(X,Course), takes(Y,Course).

```
?- classmates(tom,mary).
true .

?- classmates(tom,billy).
false.

?- classmates(tom,bob).
false.
```

The query works as tom and mary take the same course and they are both students. It shows that tom and billy don't take the same course as billy doesn't exist. Hence, it returns false. It also demonstrates the check if they are students as bob is in the same course as tom, but is a teacher. These queries demonstrate my classmate rule works as it queries if 2 people take the same course and are both students.

Complete Question 1 Code:

```
takes(tom, ct331).
takes(mary, ct331).
takes(joe, ct331).
takes(tom, ct345).
takes(mary, ct345).
instructs(bob, ct331).
instructs(ann, ct345).
teaches(X,Y) :- instructs(X,Course), takes(Y,Course).
classmates(X,Y) :- takes(X,Course), takes(Y,Course).
```

Question 2

1. **Using the "=" sign and the prolog list syntax to explicitly unify variables with parts of a list, write a prolog query that displays the head and tail of the list [1,2,3].**
   [H|T] = [1,2,3].
   ```
   ?- [H|T] = [1,2,3].
   H = 1,
   T = [2, 3].
   ```
2. **Similarly, use a nested list to display the head of the list, the head of the tail of the list and the tail of the tail of the list [1 ,2,3,4,5].**
   [H|[HT|T]] = [1,2,3,4,5].
   ```
   ?- [H|[HT|T]] = [1,2,3,4,5].
   H = 1,
   HT = 2,
   T = [3, 4, 5].
   ```
3. **Write a prolog rule 'contains1' that returns true if a given element is the first element of a given list.**
   contains1(El,[El| _]).

```
?-
|    contains1(1, [1,2,3]).
true.

?- contains1(2, [1,2,3]).
false.

?- contains1(4, [1,2,3]).
false.
```

4. **Write a prolog rule 'contains2' that returns true if a given list is the same as the tail of another given list.**

```
contains2(El,[_ |T]) :- El = T.
```

```
?- contains2([2,3], [1,2,3]).
true.

?- contains2([2,5], [1,2,3]).
false.
```

5. **Write a prolog query using 'contains1' to display the first element of a given list.**

```
contains1(Element1, [1,2,3]).
```

```
?- contains1(Element1, [1,2,3]).
Element1 = 1.

?- contains1(Element1, [5,56,34]).
Element1 = 5.
```

Question 3

```
isNotElementInList(_, []).

isNotElementInList(El, [H|T]) :-
   El\= H,
   isNotElementInList(El,T).
```

```
?-
|    isNotElementInList(1, []).
true
Unknown action: 0 (h for help)
Action? ,

?- isNotElementInList(1, []).
true ,

?- isNotElementInList(1, [1]).
false.

?-
|    isNotElementInList(1, [2]).
true ,

?- isNotElementInList(2, [1,2,3]).
false.

?-
|    isNotElementInList(7, [1,2,9,4,5]).
true ,
```

Question 4

```
mergeLists([], [], L3, L3).

mergeLists([], [H2|T2], L3, [H|T]) :-
   H = H2,
   mergeLists([], T2, L3, T).

mergeLists([H1|T1], L2, L3, [H|T]) :-
   H = H1,
   mergeLists(T1, L2, L3, T).

mergeLists(L1, L2, L3, Res) :-
   Res = [H|T],
   mergeLists(L1, L2, L3, T),
   H = L1.
```

```
?- mergeLists([7],[1,2,3],[6,7,8], X).
X = [7, 1, 2, 3, 6, 7, 8] .

?- mergeLists([2], [1], [0], X).
X = [2, 1, 0] .

?- mergeLists([1], [], [], X).
X = [1] .
```

Question 5

```
reverseList(List, R):-
   reverse(List, [], R).

reverse([ ],R,R).

reverse([H|T], Temp, R) :-
   reverse(T, [H|Temp], R).
```

```
?- reverseList([1,2,3,5,67,8],X).
X = [8, 67, 5, 3, 2, 1].

?- reverseList([],X).
X = [].

?- reverseList([1],X).
X = [1].

?- reverseList([1,2,3],X).
X = [3, 2, 1].
```

Question 6

```
insertInOrder(El, [], [El]).

insertInOrder(El, [H|T], [El, H|T]) :-
   El =< H.

insertInOrder(El, [H|T], [H|Temp]) :-
   El > H,
   insertInOrder(El, T, Temp).
```

```
?- insertInOrder(7,[1,2,3],X).
X = [1, 2, 3, 7] .

?- insertInOrder(7,[1,2,3,8],X).
X = [1, 2, 3, 7, 8] .

?- insertInOrder(1, [], X).
X = [1] .
```