11010170 Prolog Assignment 2

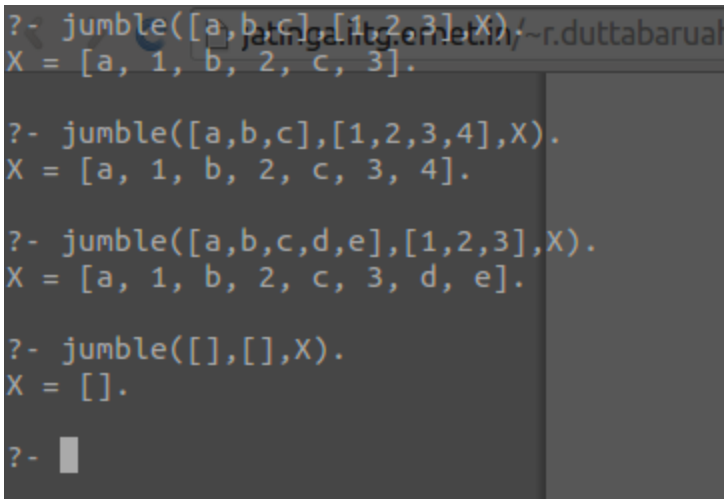
Question 1

Write a predicate ‘jumble(List1, List2, JumbledList)’ which takes three lists as arguments and returns a third list as follows:
?-jumble([a, b, c], [1,2,3], X).
X = [a, 1, b, 2, c, 3]

Solution

```
1  /*When the lists are of different sizes, these two statements take care of the case*/
2  jumble([X|L1],[],[X|L1]).
3  jumble([],[Y|L2],[Y|L2]).
4
5  /*When the lists have one each elements, the resultant list has the two elements clubbed together*/
6  jumble([X],[Y],[X,Y]).
7
8  /*Calls the jumble function with the remainder of the lists after clubbing the first elemnets of the lists*/
9  jumble([X|L1],[Y|L2],[X,Y|L3]):-jumble(L1,L2,L3),!.
10
11 /*When both the lists are empty, the output list is also empty*/
12 jumble([],[],[]).
```

Execution



Question 2

Write a predicate ‘split(Numbers, Positives, Negatives)’ which splits a list of numbers into two lists: positive ones (including zero) and negative ones. For example:

?-split([2, -1, 0, 7, -8], X, Y).

X = [2, 0, 7]

Y = [-1, -2]

Solution

(Without using cut)

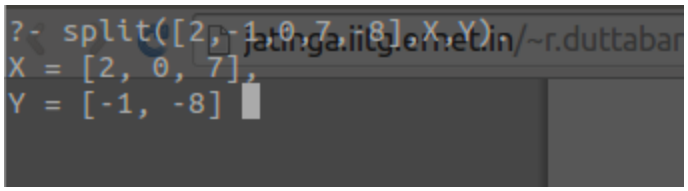
```
/*When the list is empty, the resultant list is empty as well*/
split([],[],[]).

/*When the first element is positive or zero, it is prepended to the first output list */
/*and the remaining of the resultant list is obtained recursively from the remainder of the list*/
split([X|L],[X|L1],L2):-X>=0,split(L,L1,L2).

/*When the first element is negative, it is prepended to the second output list and the remaining */
/*of the resultant list is obtained recursively from the remainder of the list*/
split([X|L],L1,[X|L2]):-X<0,split(L,L1,L2).
```

(Using cut)

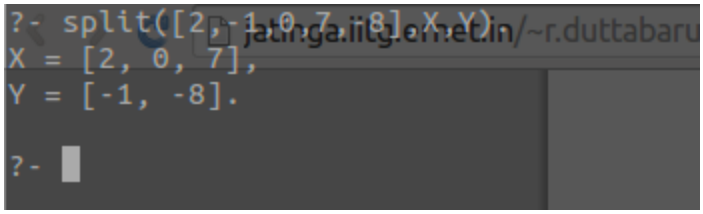
```
/*When the list is empty, the resultant list is empty as well*/
split([],[],[]).

/*When the first element is positive or zero, it is prepended to the first output list */
/*and the remaining of the resultant list is obtained recursively from the remainder of the list*/
/*The cut is used because, if the condition is true, the X need not be checked for the <0 condition as it is already >=0*/
split([X|L],[X|L1],L2):-X>=0,split(L,L1,L2),!.

/*When the first element is negative, it is prepended to the second output list */
/*and the remaining of the resultant list is obtained recursively from the remainder of the list*/
split([X|L],L1,[X|L2]):-X<0,split(L,L1,L2).
```

Execution

(Without using cut)



(Using cut)



Question 3

Write a predicate nu (for not unifiable) which takes two terms as arguments and succeeds if the two terms do not unify.

(a) Write it with the help of = and \+.

(b) Write it using cut-fail combination and don't use = and +\.

Solution

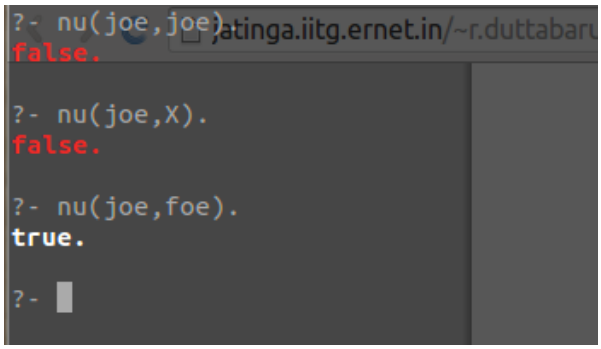
(Without using cut-fail) Part a

```
1  /*The boolean value of comaprison of X and Y values are evaluated and negated*/
2  nu(X,Y):- \+ X=Y.
```

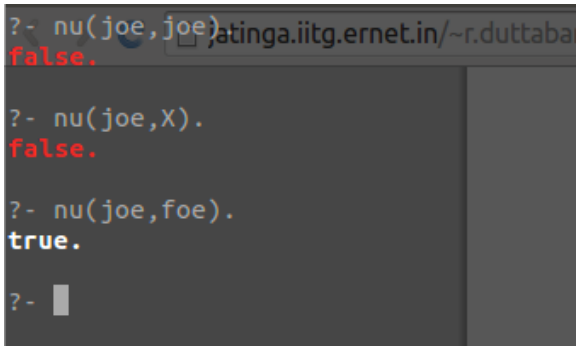
(Using cut-fail) Part b

```
1  /*When the elements are the same, it prevents backtracking by using cut and the fail statement makes it return false*/
2  nu(X,X) :- !,fail.
3
4  /*When the control reaches here, the wildcard entries are different here and the cut */
5  /*prevents backtracking to the first statement and it returns true*/
6  nu(_,_) :- !.
```

Execution



(Without using cut-fail) Part a



(Using cut-fail) Part b