Project- Artificial Intelligence

Title:

**Create linked lists using recursive structures**

**Explain of the program:**

Linked list has two components, the integer part and the link part. The link part will hold another node. End of list will have nil into the link part.

To express this so will use **node (2, node(5, node(6, nil)))** in our prolog:

**Main steps to consider:**

* The smallest possible list is nil, and every other list will contain nil as the "next" of the end node.
* In list terminology, the first element is usually called the **head** of the list, and the rest of the list is called the **tail** part.
* The head of the above list is 2, and its tail is the list node (5, node (6, nil)).
* We can also insert elements into front and back side

The code:

**add\_front(L,E,NList) :- NList = node(E,L).**

**add\_back(nil, E, NList) :-**

**NList = node(E,nil).**

**add\_back(node(Head,Tail), E, NList) :-**

**add\_back(Tail, E, NewTail),**

**NList = node(Head,NewTail).**

The output should bs like this:

| ?- [linked\_list].

compiling D:/TP Prolog/Sample\_Codes/linked\_list.pl for byte code...

D:/TP Prolog/Sample\_Codes/linked\_list.pl compiled, 7 lines read - 966 bytes written, 14 ms

(15 ms) yes

| ?- add\_front(nil, 6, L1), add front(L1, 5, L2), add\_front(L2, 2, L3).

L1 = node(6,nil)

L2 = node(5,node(6,nil))

L3 = node(2,node(5,node(6,nil)))

yes

| ?- add\_back(nil, 6, L1), add\_back(L1, 5, L2), add\_back(L2, 2, L3).

L1 = node(6,nil)

L2 = node(6,node(5,nil))

L3 = node(6,node(5,node(2,nil)))

yes

| ?- add\_front(nil, 6, L1), add\_front(L1, 5, L2), add\_back(L2, 2, L3).

L1 = node(6,nil)

L2 = node(5,node(6,nil))

L3 = node(5,node(6,node(2,nil)))

yes

| ?-

Screenshot of program:



Screenshot of output:

