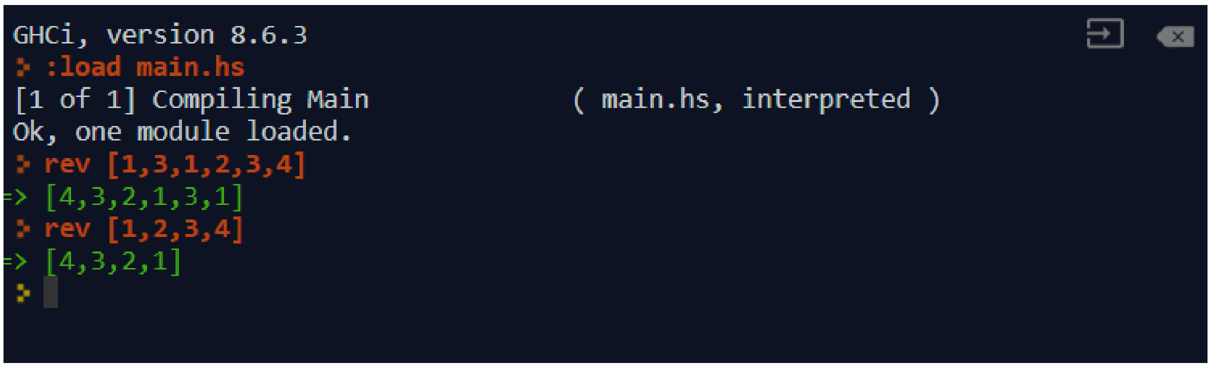
Himanish Pathak/hp1235

1. Write a program in each pf the following languages: Haskell, Python, Prolog and Scheme that reverses a list of integers. Do not use higher order functions such as "reverse". Use recursion.

<u>Haskell:</u>

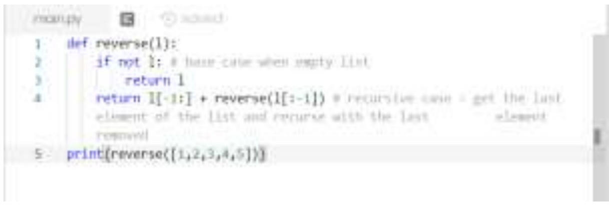
```haskell
rev :: [a] -> [a]
rev [] = []
rev [x] = [x]
rev (x:xs) = rev xs ++ [x]
```



```
GHCi, version 8.6.3
> :load main.hs
[1 of 1] Compiling Main             ( main.hs, interpreted )
Ok, one module loaded.
> rev [1,3,1,2,3,4]
=> [4,3,2,1,3,1]
> rev [1,2,3,4]
=> [4,3,2,1]
>
```

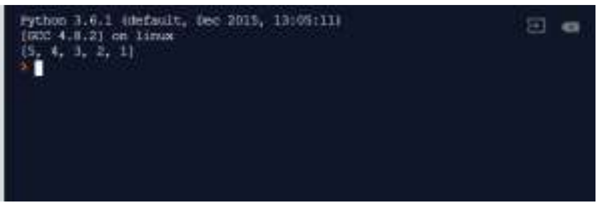
<u>Python:</u>

```python
def reverse(l):
    if not l: # base case when empty list
        return l
    return l[-1:] + reverse(l[:-1]) # recursive case : get the last element of
the list and recurse with the last          element removed
print(reverse([1,2,3,4,5]))
```



main.py

```python
def reverse(l):
    if not l: # base case when empty list
        return l
    return l[-1:] + reverse(l[:-1]) # recursive case : get the last
    element of the list and recurse with the last          element
    removed
print(reverse([1,2,3,4,5]))
```

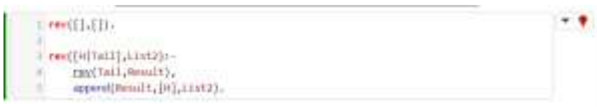


```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
[5, 4, 3, 2, 1]
>
```

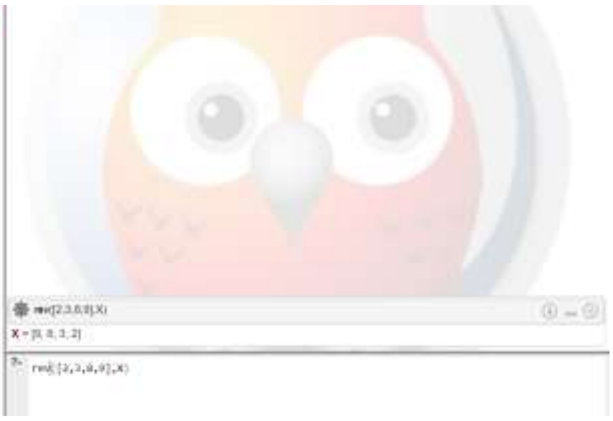
Himanish Pathak/hp1235

<u>Prolog:</u>

```
append([],L,L).

append([H|Tail],List2,[H|List3]) :-

        append(Tail,List2,List3).

rev([],[]).

rev([H|Tail],List2):-

        rev(Tail,Result),
        append(Result,[H],List2).
```



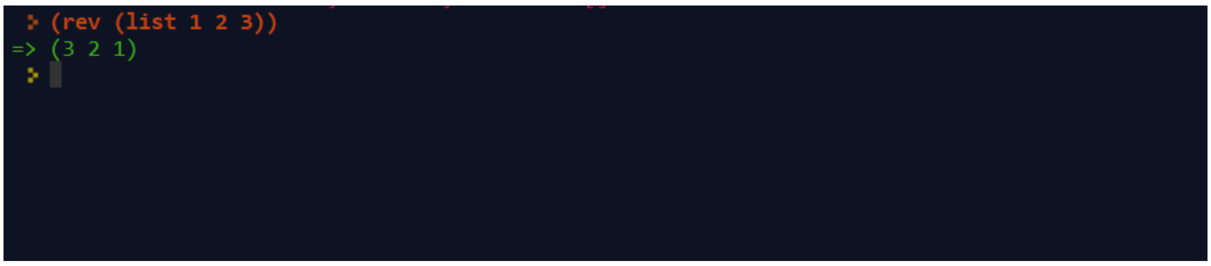
```
rev([],[]).

rev([H|Tail],List2):-
    rev(Tail,Result),
    append(Result,[H],List2).
```



```
rev([2,3,8,8],X)
X = [8, 8, 3, 2]
?- rev([2,3,8,8],X)
```

<u>Scheme:</u>

```
(define (rev ls)
  (myrev ls ()))

(define (myrev ls0 ls1)
  (if (null? ls0)
      ls1
      (myrev (cdr ls0) (cons (car ls0) ls1))))
```



```
> (rev (list 1 2 3))
=> (3 2 1)
>
```