IEEE NITK
Functional Programming with Erlang
ASSIGNMENT - 4

Deadline for submission : 21 June 2017 by 11 : 59 pm

1. Implement a for loop construct in Erlang as a function. DO NOT use any other variables other than the parameters I, Pred, Update and Body.

for(I, Pred, Update, Body) ->

Example usage:
A for loop to print 0 to 9:

```
for(0, fun(X) -> X < 10 end, fun(X) -> X+1 end, fun(X) ->
 io:format("~i~n", [X]) end).
```

Notice that Pred, Update and Body are all unary functions.

2. It is sometimes useful to know where in a list a certain element occurs, if it occurs at all. Program the function index-in-list-by-predicate which searches for a given element. The comparison between the given element and the elements in the list is controlled by a comparison parameter to index-in-list-by-predicate. The function should return the list **position** of match (first element is number 0), or the atom **not_found** if no match is found.

Example usage:

```
index-in-list-by-predicate([20,10,30,40], 10, f(X,Y) ->
 X =:= Y end).
% should return 1
index-in-list-by-predicate([20,10,30,40], 50, f(X,Y) ->
 X =:= Y end).
% should return not_found
index-in-list-by-predicate([20,10,3,40], 20, f(X,Y) ->
 X-Y =:= 17 end).
% should return 2
```

3. The mathematical quantifiers **for all** and **there exists** are well-known. Implement the following three Boolean functions (should return either true or false):

- The function **for-all(List, Pred)** is supposed to check if all elements in the list List satisfy the predicate Pred.

- The function **there-exists(List, Pred)** is supposed to check if one or more elements in the list List satisfy the predicate Pred.

- Finally, the function **there-exists-2(List, Pred)** is supposed to check if exactly two elements in the list List satisfy Pred.

4. The functions foldr, and foldl are fold equivalents, except that in foldr, we start processing from end of the list (hence fold right, or foldr), and in foldl, we start processing from beginning of the list (hence fold left, or foldl).

- Implement foldr, foldl.
- Implement filterr using foldr, and filterl using foldl.

5. Functions scanl and scanr are like foldl and foldr, only they report all the intermediate accumulator states in the form of a list. When using a scanl, the final result will be in the last element of the resulting list while a scanr will place the result in the head.
Implement **scanl and scanr** which return a list.

6. The function rem_dups removes adjacent duplicates from a list.

```
For example,
remdups ([1, 2, 2, 3, 3, 3, 1, 1]).
% should return [1, 2, 3, 1].
```

- Define rem_dups using foldr .
- Give another definition using foldl.

7. Implement quick sort using **filterr/filterl** to filter out the list elements that are smaller than (or equal to) and larger than the pivot.

8. Write a function that takes a list of even numbers as argument and returns sum of squares of all numbers which can be expressed as sum of 2 primes. Use foldl/foldr.

9. The function **takewhile(Pred,List)** takes elements Elem from List while Pred(Elem) returns true, that is, the function returns the longest prefix of the list for which all elements satisfy the predicate.

Refer : http://erlang.org/doc/man/lists.html#takewhile-2

- Implement your own version of takewhile function.

- How many elements does it take for the sum of the square roots of all natural numbers to exceed 1000? Answer this question with a program making use of **scanl and takewhile** functions.

10. Write a function making use of foldr/foldl which takes a list of n positive integers and another integer s as arguments and returns the **size** of minimal subset whose sum is greater than or equal to s. If there exists no such subset then return the atom **not_found** instead.

Instructions :

- Wherever needed, try to check the performance of your functions using the timer function.
- For submission, follow the same instructions as previous assignments.
- All your files should go in **Week 5/Assignment 3/<your_name>**