Eddie Gurnee
E01020445
COSC 341
4/14/2015

COSC 341
Project 7

Erlang: Once Upon A Haskell

```erlang
%% @author eddie
%% @doc @todo Add description to erlang_mini_project.

-module(erlang_mini_project).
-author(eddie).

%% ====================================================================
%% API functions
%% ====================================================================
-export([count/2, substring/2, get_last/1, zip/2, permute/1, ackermann/2]).

count(_, []) -> 0;
count(X, [Y|YS]) ->
    case X =:= Y of
        true -> 1 + count(X, YS);
        false -> count(X, YS)
    end.

substring(X,Y) -> subHelp(0, X, Y).

get_last([]) -> [];
get_last([X]) -> X;
get_last([_, Y|Tail]) -> get_last([Y|Tail]).

zip([], _) -> [];
zip(_, []) -> [];
zip([Head1|Tail1], [Head2|Tail2]) -> [{Head1, Head2} | zip(Tail1, Tail2)].

permute([]) -> [[]];
permute(List) -> [[Head|Tail] || Head <- List, Tail <- permute(List -- [Head])].

ackermann(M, N) when M < 0; N < 0 -> fail_blog;
ackermann(0, N) -> N + 1;
ackermann(M, 0) -> ackermann(M - 1, 1);
ackermann(M, N) -> ackermann(M - 1, ackermann(M, N - 1)).

%% ====================================================================
%% Internal functions
%% ====================================================================

%% This implementation of substring only came to fruition through discussing
%  theory with Holice Kil. That man is a genius.
subHelp(_, _, []) -> -1;
subHelp(Index, X, Y) ->
    case take(Y, length(X)) of
        X -> Index;
        _ -> subHelp(Index + 1, X, tail(Y))
    end.

take([], _) -> [];
take(_, 0) -> [];
take([Head|Tail], N) -> [Head|take(Tail, N - 1)].

tail([]) -> [];
tail([_|T]) -> T.
```