University of Kent

Introduction … where do I begin?

Where do I begin?

Look at a collection of strategies.
Work through practical examples … in real time.

Example #1: `take`

Define a function `take` that takes the first `N` elements from a list.

Example #1: examples of `take` in use

Define a function `take` that takes the first `N` elements from a list.

```
take(0,"hello") = []
take(4,"hello") = "hell"
take(5,"hello") = "hello"
take(9,"hello") = "hello"
```

Example #1: what is its type?

Define a function **take** that takes the first **N** elements from a list.

```
-spec take(integer(),[T]) -> [T].
```

type of the result

the function take has 2 arguments

type of the first argument

type of the 2nd argument

Example #1: Doing it ourselves ...

Let's define it for ourselves. We have templates for integers,

```
foo(0) ->
  ... ;
foo(N) when N>0 ->
  ... foo(N-1) ... .
```

and templates for lists

```
bar([]) ->
    ... ;
bar([X|Xs]) ->
    ... bar(Xs) ... .
```

We have both lists and integers here ...

which should we think of using?

Example #1: examples of `take` in use

Define a function **take** that takes the first **N** elements from a list.

```
take(0,"hello") = []
take(4,"hello") = "hell"
take(5,"hello") = "hello"
take(9,"hello") = "hello"
```

Terminal Shell Edit View Window Help
University of Kent
simonthompson — xterm — beam.smp
where1.erl
New Open Recent Revert Save Print Undo Redo Cut Copy Paste Search Preferences Help

```
Last login: Tue May  5 09:48:45 on console
% bash; exit
% ,
bash: ,: command not found
% ./bin/ba
bash: ./bin/ba: No such file or directory
% ./bin/backup
%
  [Restored]
Last login: Tue May  5 10:05:40 on ttys000
% bash; exit
% cd Desktop/
% erl
Erlang/OTP 17 [erts-6.3] [source-f9282c6] [64-bit] [smp:8:8] [async-threa
ds:10] [hipe] [kernel-poll:false]

Eshell V6.3  (abort with ^G)
1> c(where1).
{ok,where1}
2> where1:take(0,"hello").
[]
3> where1:take(4,"hello").
"hell"
4> where1:take(5,"hello").
"hello"
5> where1:take(9,"hello").
"hello"
6> 
```

```erlang
-module(where1).
-export([take/2]).

take(0,_Xs) ->
    [];
take(_N,[]) ->
    [];
take(N,[X|Xs]) when N>0 ->
    [X|take(N-1,Xs)].
```

```
U:---  where1.erl    All (2,15)    (Erlang EXT Flymake)
Wrote /Users/simonthompson/Desktop/where1.erl
```

Example #1: can we reuse something?

Let's take a look in the `lists` module …

```
lists:split(0,"hello") = {[],"hello"}
lists:split(4,"hello") = {"hell","o"}
```

`lists:split(9,"hello")` gives an error.

Example #1: reuse the function …

We can use the **`lists:split`** function itself …

```
take(N,Xs) ->
  {Front,_Back} = lists:split(N,Xs),
  Front.
```

Example #1: … or reuse the definition

… or modify the definition of `split` in the module `lists.erl`

```
split(N, List) ->
       split(N, List, []).

split(0, L, R) ->
       {lists:reverse(R, []), L};
split(N, [H|T], R) ->
       split(N-1, T, [H|R]);
split(_, [], _) ->
       badarg.
```

This is tail recursive - and therefore a more complicated but more efficient definition.

Example #1: … or reuse the definition

… or modify the definition of `split` in the module `lists.erl`

```
split(N, List) ->
        split(N, List, []).

split(0, L, R) ->
        {lists:reverse(R, []), L};
split(N, [H|T], R) ->
        split(N-1, T, [H|R]);
split(_, [], _) ->
        badarg.
```

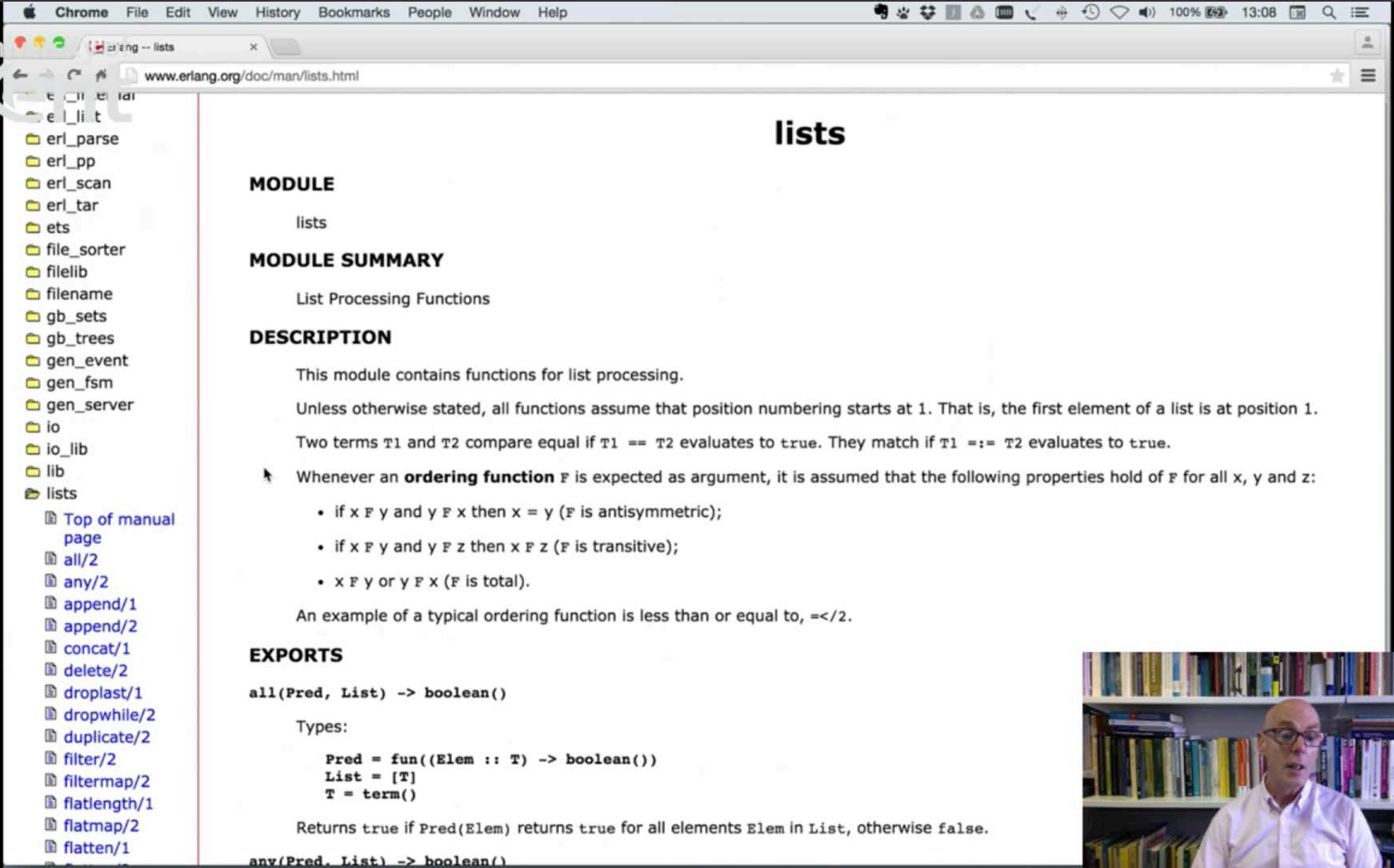
```
take(N, List) ->
        take(N, List, []).

take(0, L, R) ->
        lists:reverse(R, []);
take(N, [H|T], R) ->
        take(N-1, T, [H|R]);
take(_, [], _) ->
        badarg.
```

Example #1: … or reuse the definition

… or modify the definition of **split** in the module **lists.erl**

```
split(N, List) ->
      split(N, List, []).

split(0, L, R) ->
       {lists:reverse(R, []), L};
split(N, [H|T], R) ->
       split(N-1, T, [H|R]);
split(_, [], _) ->
       badarg.
```

```
take(N, List) ->
      take(N, List, []).

take(0, L, R) ->
       lists:reverse(R, []);
take(N, [H|T], R) ->
       take(N-1, T, [H|R]);
take(_, [], R) ->
       lists:reverse(R,[]).
```


Chrome File Edit View History Bookmarks People Window Help
100% 13:08
erlang -- lists
www.erlang.org/doc/man/lists.html

- erl_parse
- erl_pp
- erl_scan
- erl_tar
- ets
- file_sorter
- filelib
- filename
- gb_sets
- gb_trees
- gen_event
- gen_fsm
- gen_server
- io
- io_lib
- lib
- lists
  - Top of manual page
  - all/2
  - any/2
  - append/1
  - append/2
  - concat/1
  - delete/2
  - droplast/1
  - dropwhile/2
  - duplicate/2
  - filter/2
  - filtermap/2
  - flatlength/1
  - flatmap/2
  - flatten/1

lists

MODULE

lists

MODULE SUMMARY

List Processing Functions

DESCRIPTION

This module contains functions for list processing.

Unless otherwise stated, all functions assume that position numbering starts at 1. That is, the first element of a list is at position 1.

Two terms `T1` and `T2` compare equal if `T1 == T2` evaluates to `true`. They match if `T1 =:= T2` evaluates to `true`.

Whenever an **ordering function** `F` is expected as argument, it is assumed that the following properties hold of `F` for all x, y and z:

- if x `F` y and y `F` x then x = y (`F` is antisymmetric);
- if x `F` y and y `F` z then x `F` z (`F` is transitive);
- x `F` y or y `F` x (`F` is total).

An example of a typical ordering function is less than or equal to, `=</2`.

EXPORTS

`all(Pred, List) -> boolean()`

Types:

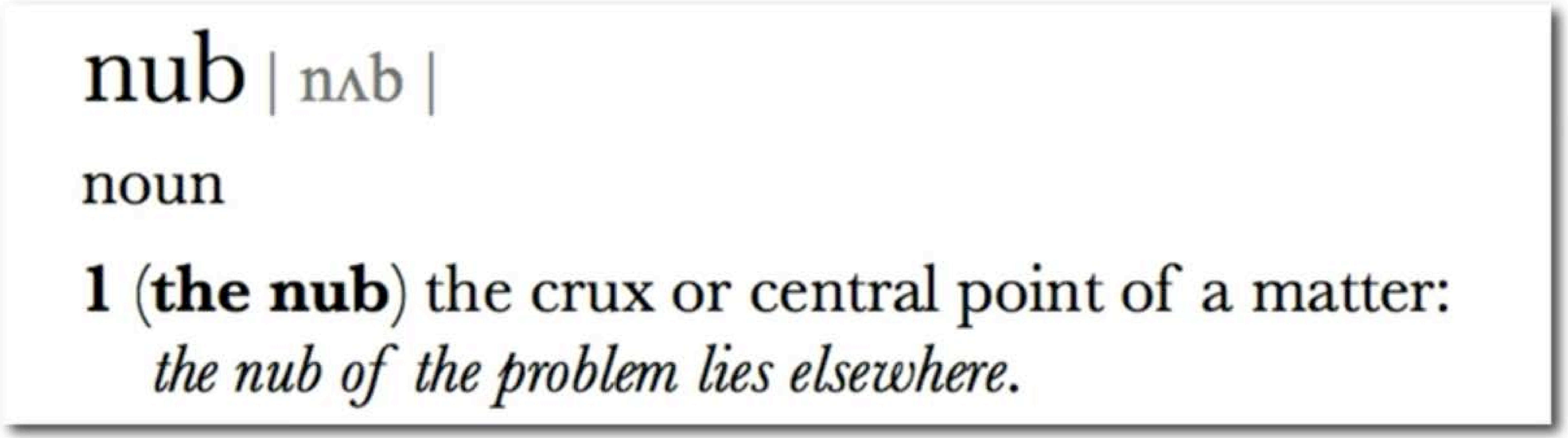
```
Pred = fun((Elem :: T) -> boolean())
List = [T]
T = term()
```

Returns `true` if `Pred(Elem)` returns `true` for all elements `Elem` in `List`, otherwise `false`.

Where do I begin? `nub`

University of Kent

Example #2: nub



nub | nʌb |

noun

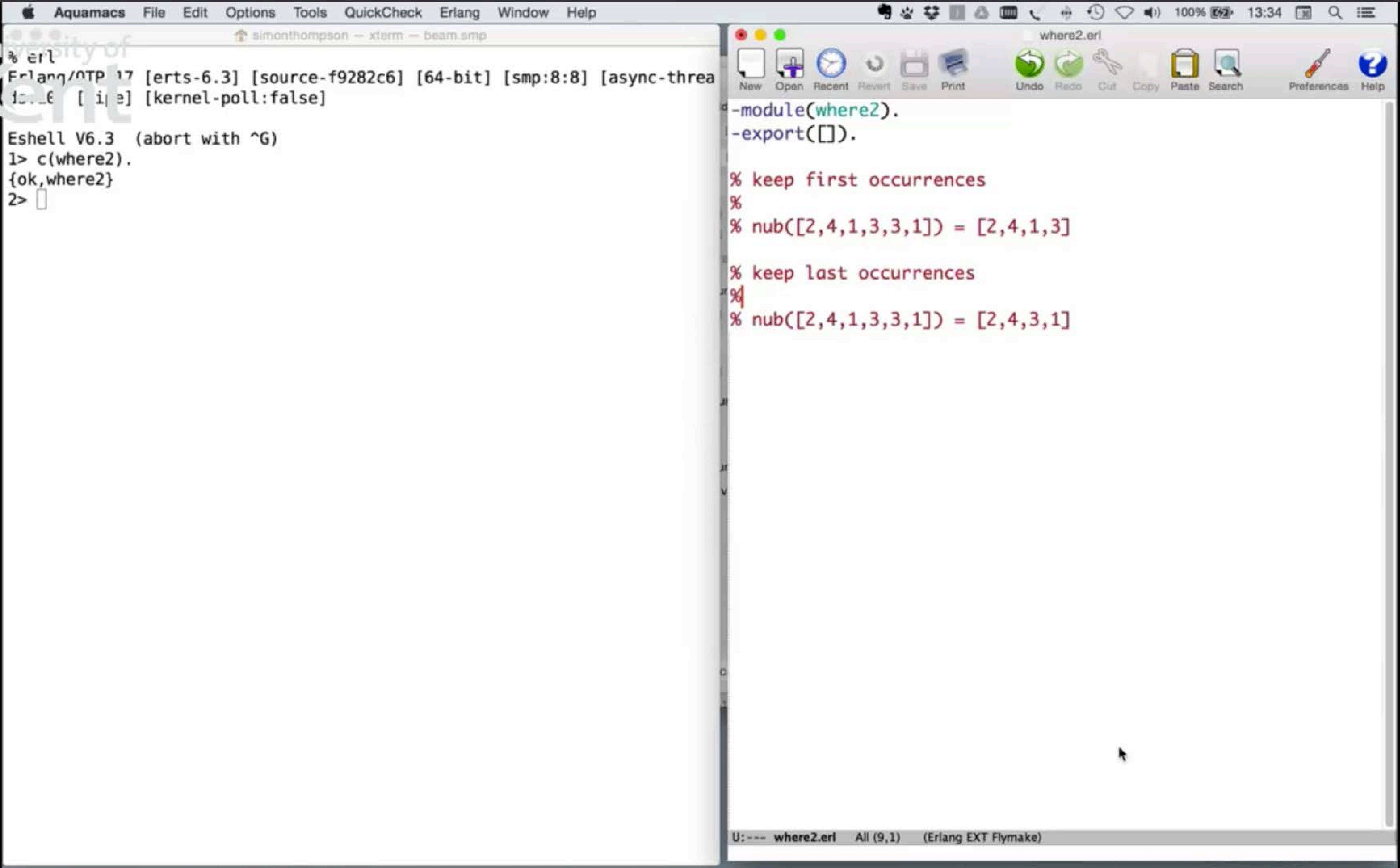
1 (**the nub**) the crux or central point of a matter:

the nub of the problem lies elsewhere.

Example #2: `nub`

Remove all the duplicate elements from a list.

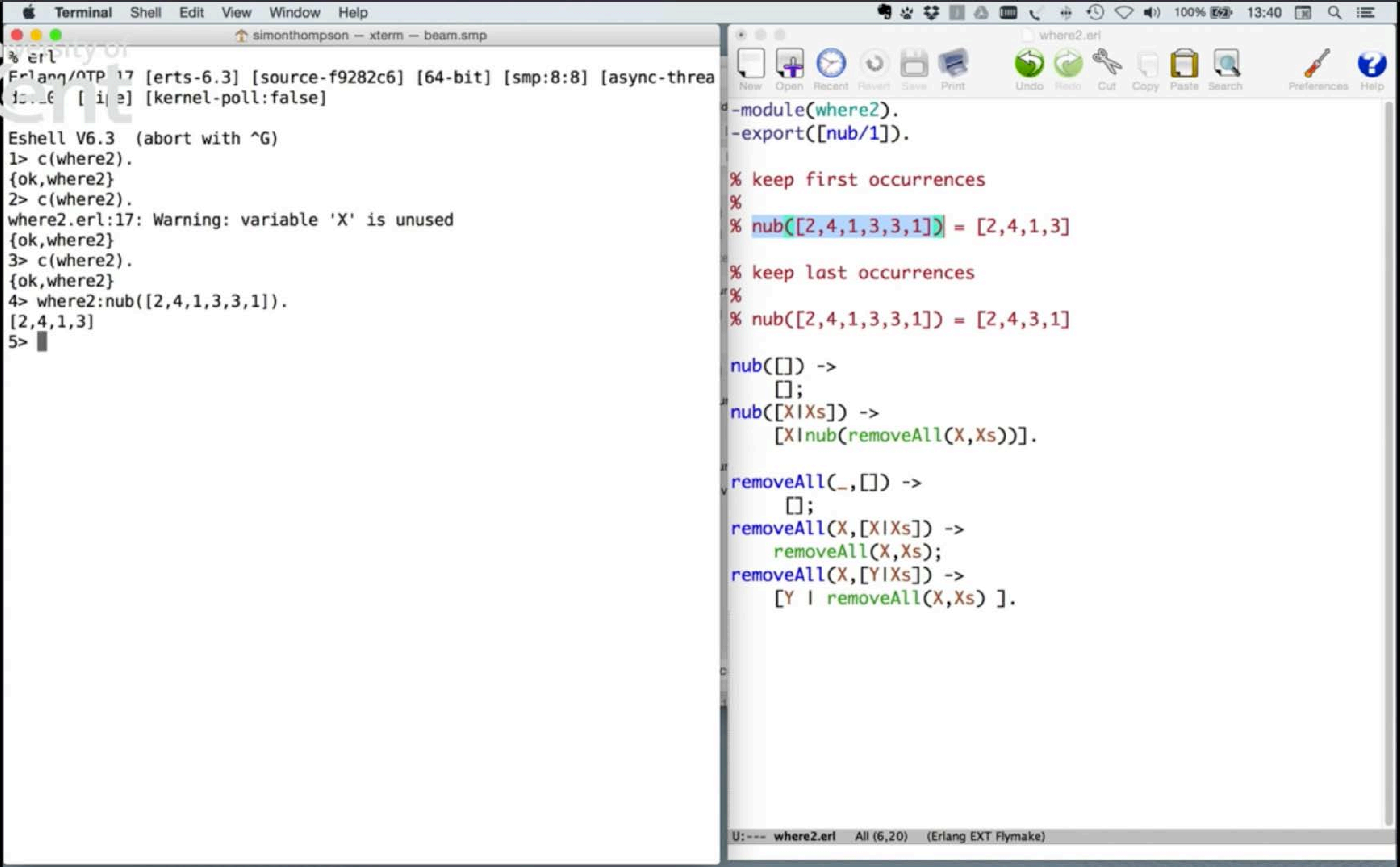
```
nub([2,4,1,3,3,1]) = [2,4,1,3]
nub([2,4,1,3,3,1]) = [2,4,3,1]
```


Aquamacs File Edit Options Tools QuickCheck Erlang Window Help

```
% erl
Erlang/OTP 17 [erts-6.3] [source-f9282c6] [64-bit] [smp:8:8] [async-threa
ds:10] [hipe] [kernel-poll:false]

Eshell V6.3  (abort with ^G)
1> c(where2).
{ok,where2}
2> 
```

```
-module(where2).
-export([]).

% keep first occurrences
%
% nub([2,4,1,3,3,1]) = [2,4,1,3]

% keep last occurrences
%
% nub([2,4,1,3,3,1]) = [2,4,3,1]
```

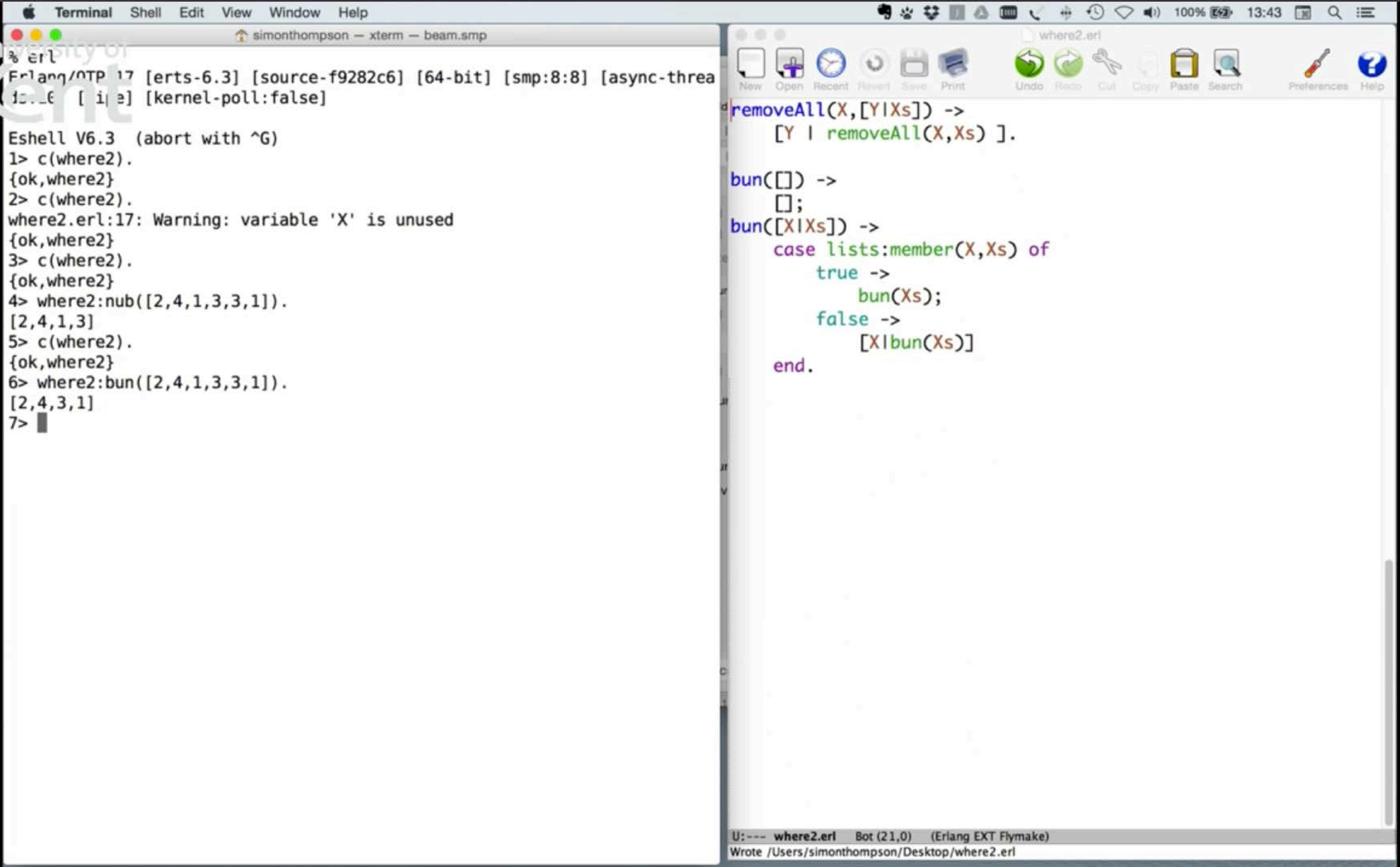
U:--- where2.erl All (9,1) (Erlang EXT Flymake)


Terminal Shell Edit View Window Help
simonthompson — xterm — beam.smp
where2.erl
New Open Recent Revert Save Print Undo Redo Cut Copy Paste Search Preferences Help

```
% erl
Erlang/OTP 17 [erts-6.3] [source-f9282c6] [64-bit] [smp:8:8] [async-threa
ds:10] [hipe] [kernel-poll:false]

Eshell V6.3  (abort with ^G)
1> c(where2).
{ok,where2}
2> c(where2).
where2.erl:17: Warning: variable 'X' is unused
{ok,where2}
3> c(where2).
{ok,where2}
4> where2:nub([2,4,1,3,3,1]).
[2,4,1,3]
5>
```

```erlang
-module(where2).
-export([nub/1]).

% keep first occurrences
%
% nub([2,4,1,3,3,1]) = [2,4,1,3]

% keep last occurrences
%
% nub([2,4,1,3,3,1]) = [2,4,3,1]

nub([]) ->
    [];
nub([X|Xs]) ->
    [X|nub(removeAll(X,Xs))].

removeAll(_,[]) ->
     [];
removeAll(X,[X|Xs]) ->
    removeAll(X,Xs);
removeAll(X,[Y|Xs]) ->
    [Y | removeAll(X,Xs) ].
```

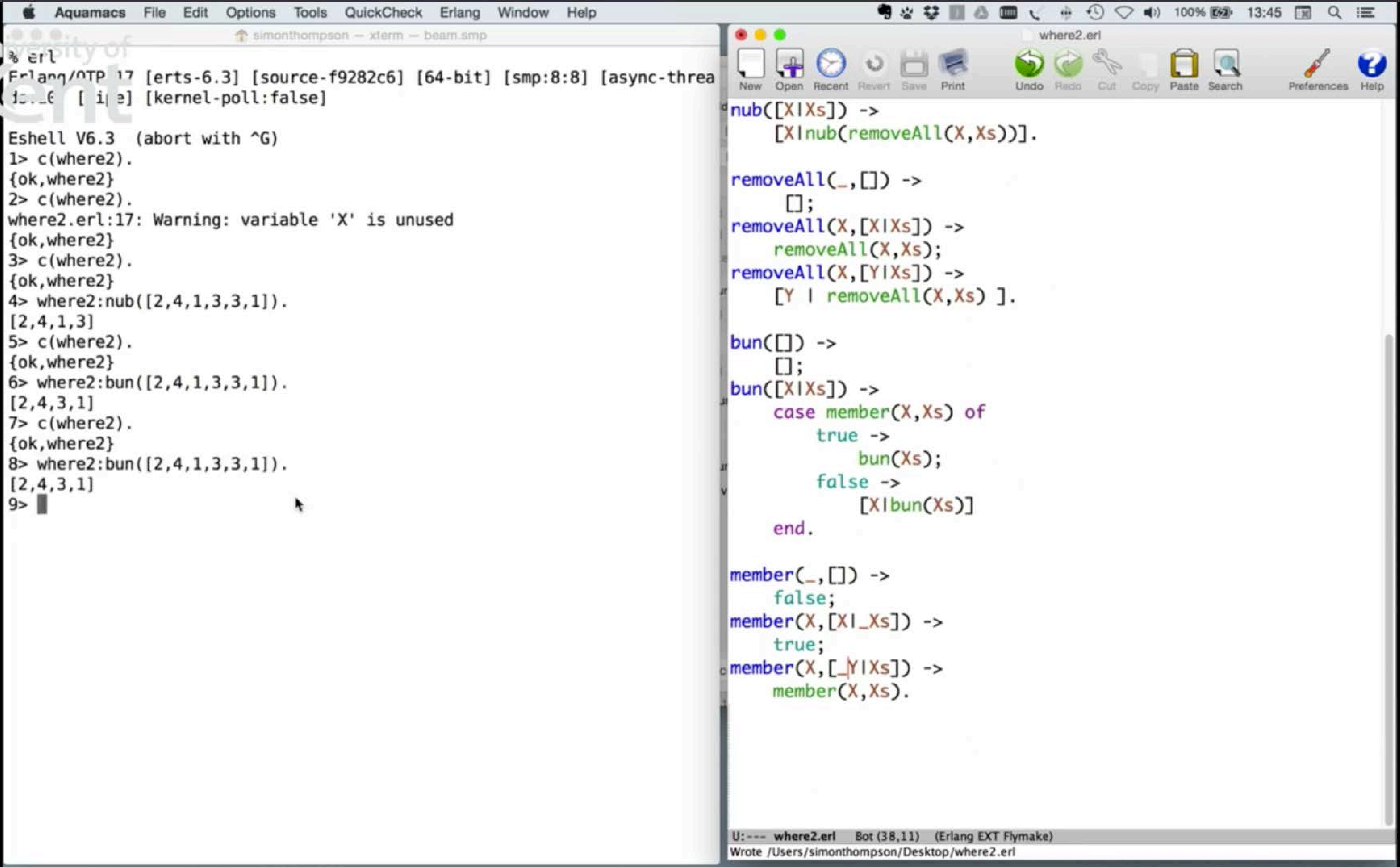
```
U:---  where2.erl    All (6,20)    (Erlang EXT Flymake)
```



```
% erl
Erlang/OTP 17 [erts-6.3] [source-f9282c6] [64-bit] [smp:8:8] [async-threa
ds:10] [hipe] [kernel-poll:false]

Eshell V6.3  (abort with ^G)
1> c(where2).
{ok,where2}
2> c(where2).
where2.erl:17: Warning: variable 'X' is unused
{ok,where2}
3> c(where2).
{ok,where2}
4> where2:nub([2,4,1,3,3,1]).
[2,4,1,3]
5> c(where2).
{ok,where2}
6> where2:bun([2,4,1,3,3,1]).
[2,4,3,1]
7> 
```

```
removeAll(X,[Y|Xs]) ->
    [Y | removeAll(X,Xs) ].

bun([]) ->
    [];
bun([X|Xs]) ->
    case lists:member(X,Xs) of
        true ->
            bun(Xs);
        false ->
            [X|bun(Xs)]
    end.
```

```
U:---  where2.erl   Bot (21,0)   (Erlang EXT Flymake)
Wrote /Users/simonthompson/Desktop/where2.erl
```


Aquamacs File Edit Options Tools QuickCheck Erlang Window Help
100% 13:45
simonthompson — xterm — beam.smp
where2.erl
New
Open
Recent
Revert
Save
Print
Undo
Redo
Cut
Copy
Paste
Search
Preferences
Help

```
% erl
Erlang/OTP 17 [erts-6.3] [source-f9282c6] [64-bit] [smp:8:8] [async-threa
ds:10] [hipe] [kernel-poll:false]

Eshell V6.3  (abort with ^G)
1> c(where2).
{ok,where2}
2> c(where2).
where2.erl:17: Warning: variable 'X' is unused
{ok,where2}
3> c(where2).
{ok,where2}
4> where2:nub([2,4,1,3,3,1]).
[2,4,1,3]
5> c(where2).
{ok,where2}
6> where2:bun([2,4,1,3,3,1]).
[2,4,3,1]
7> c(where2).
{ok,where2}
8> where2:bun([2,4,1,3,3,1]).
[2,4,3,1]
9> 
```

```erlang
nub([X|Xs]) ->
    [X|nub(removeAll(X,Xs))].

removeAll(_,[]) ->
     [];
removeAll(X,[X|Xs]) ->
    removeAll(X,Xs);
removeAll(X,[Y|Xs]) ->
    [Y | removeAll(X,Xs) ].

bun([]) ->
    [];
bun([X|Xs]) ->
    case member(X,Xs) of
        true ->
            bun(Xs);
        false ->
            [X|bun(Xs)]
    end.

member(_,[]) ->
    false;
member(X,[X|_Xs]) ->
    true;
member(X,[_Y|Xs]) ->
    member(X,Xs).
```

```
U:---  where2.erl    Bot (38,11)    (Erlang EXT Flymake)
Wrote /Users/simonthompson/Desktop/where2.erl
```

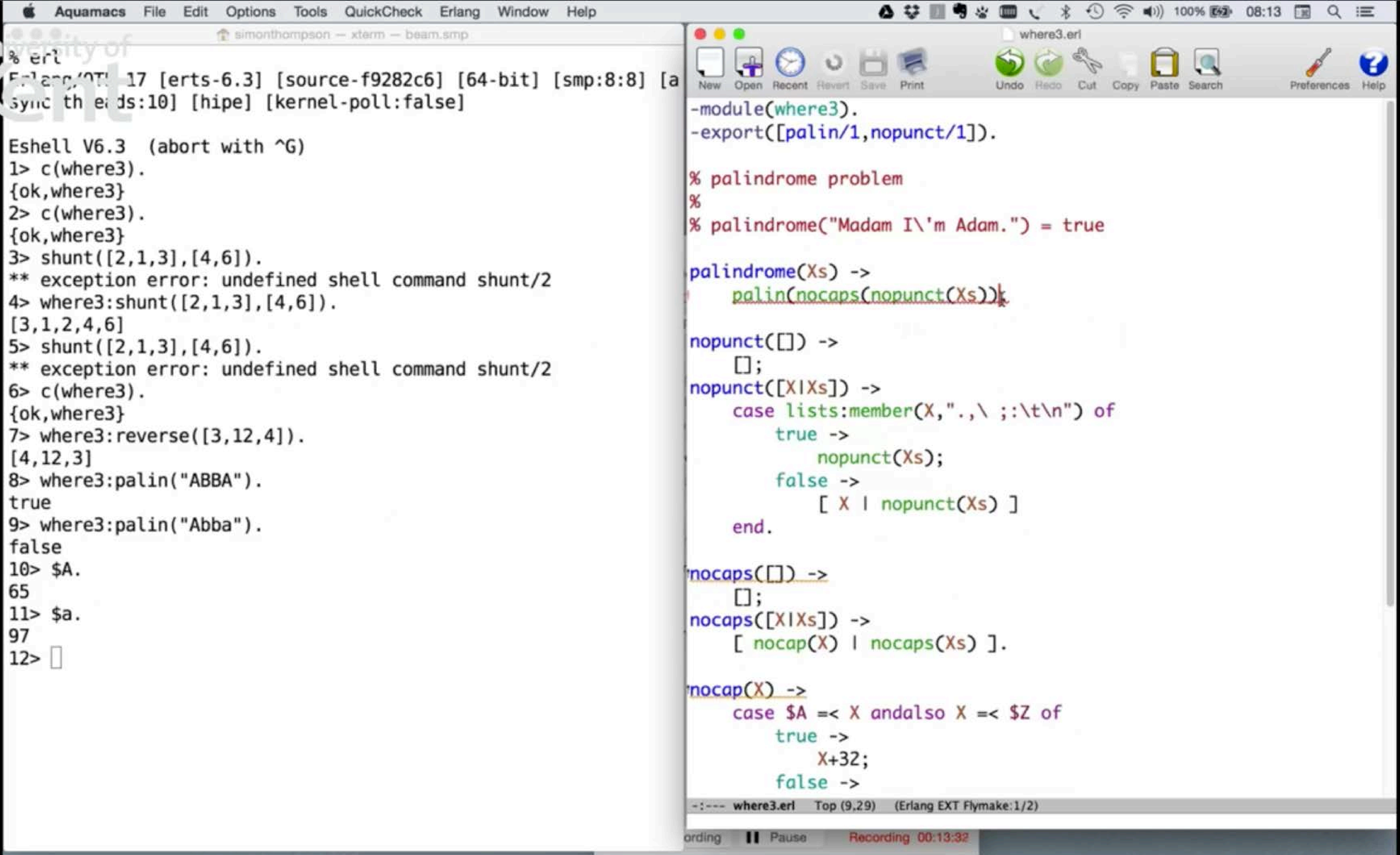
University of Kent

Where do I begin? `palindrome`

Example #3: `palindrome`

Is it a palindrome?

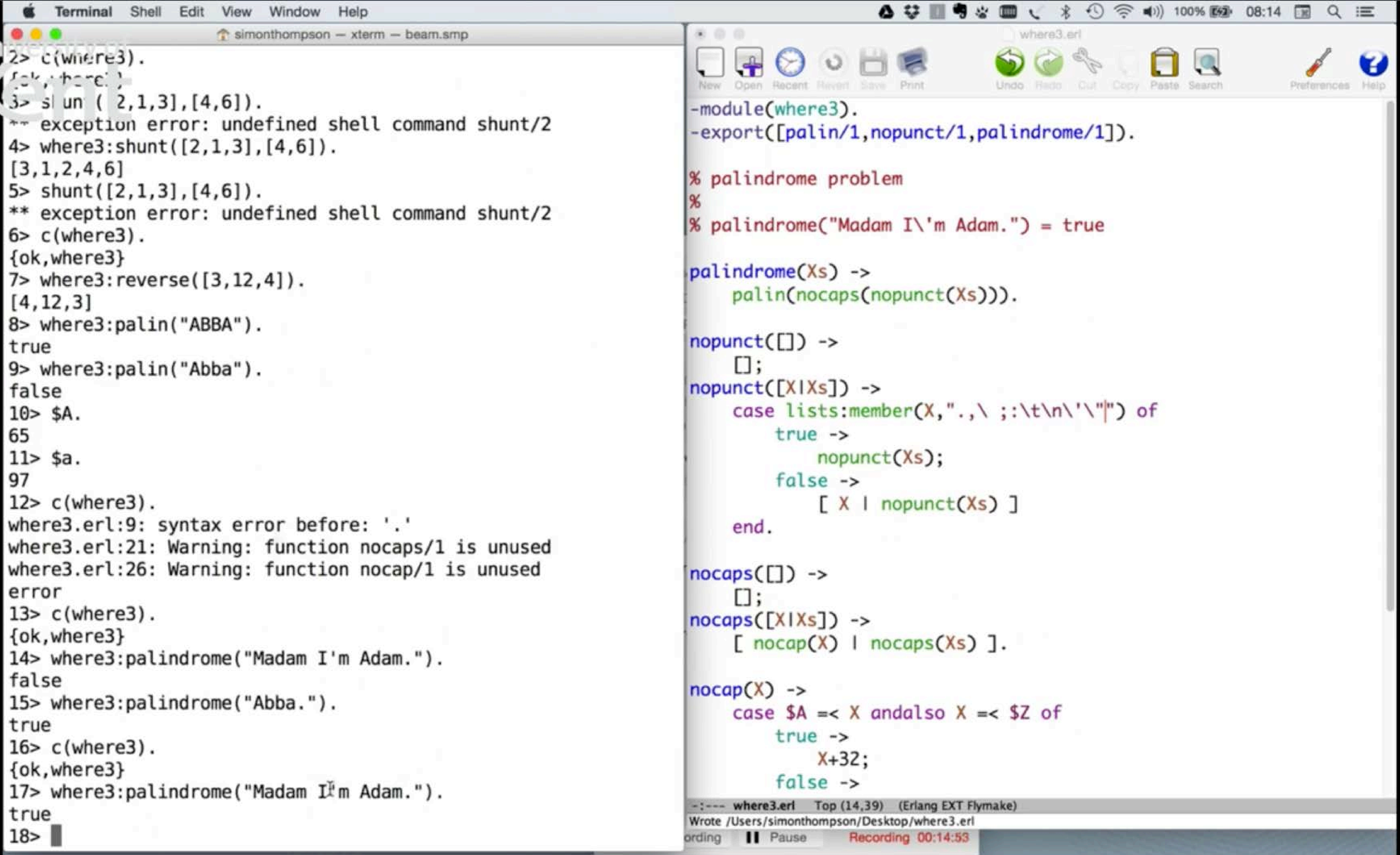
```
palindrome("Madam I\'m Adam") = true
```



```
% erl
Erlang/OTP 17 [erts-6.3] [source-f9282c6] [64-bit] [smp:8:8] [a
sync-threads:10] [hipe] [kernel-poll:false]

Eshell V6.3  (abort with ^G)
1> c(where3).
{ok,where3}
2> c(where3).
{ok,where3}
3> shunt([2,1,3],[4,6]).
** exception error: undefined shell command shunt/2
4> where3:shunt([2,1,3],[4,6]).
[3,1,2,4,6]
5> shunt([2,1,3],[4,6]).
** exception error: undefined shell command shunt/2
6> c(where3).
{ok,where3}
7> where3:reverse([3,12,4]).
[4,12,3]
8> where3:palin("ABBA").
true
9> where3:palin("Abba").
false
10> $A.
65
11> $a.
97
12> 
```

```
-module(where3).
-export([palin/1,nopunct/1]).

% palindrome problem
%
% palindrome("Madam I\'m Adam.") = true

palindrome(Xs) ->
    palin(nocaps(nopunct(Xs))).

nopunct([]) ->
    [];
nopunct([X|Xs]) ->
    case lists:member(X,".,\ ;:\t\n") of
        true ->
            nopunct(Xs);
        false ->
            [ X | nopunct(Xs) ]
    end.

nocaps([]) ->
    [];
nocaps([X|Xs]) ->
    [ nocap(X) | nocaps(Xs) ].

nocap(X) ->
    case $A =< X andalso X =< $Z of
        true ->
            X+32;
        false ->
```

```
-:--- where3.erl   Top (9,29)   (Erlang EXT Flymake:1/2)
```



```
2> c(where3).
{ok,where3}
3> shun([2,1,3],[4,6]).
** exception error: undefined shell command shunt/2
4> where3:shunt([2,1,3],[4,6]).
[3,1,2,4,6]
5> shunt([2,1,3],[4,6]).
** exception error: undefined shell command shunt/2
6> c(where3).
{ok,where3}
7> where3:reverse([3,12,4]).
[4,12,3]
8> where3:palin("ABBA").
true
9> where3:palin("Abba").
false
10> $A.
65
11> $a.
97
12> c(where3).
where3.erl:9: syntax error before: '.'
where3.erl:21: Warning: function nocaps/1 is unused
where3.erl:26: Warning: function nocap/1 is unused
error
13> c(where3).
{ok,where3}
14> where3:palindrome("Madam I'm Adam.").
false
15> where3:palindrome("Abba.").
true
16> c(where3).
{ok,where3}
17> where3:palindrome("Madam I'm Adam.").
true
18> 
```

```erlang
-module(where3).
-export([palin/1,nopunct/1,palindrome/1]).

% palindrome problem
%
% palindrome("Madam I\'m Adam.") = true

palindrome(Xs) ->
    palin(nocaps(nopunct(Xs))).

nopunct([]) ->
    [];
nopunct([X|Xs]) ->
    case lists:member(X,".,\ ;:\t\n\'\"") of
        true ->
            nopunct(Xs);
        false ->
            [ X | nopunct(Xs) ]
    end.

nocaps([]) ->
    [];
nocaps([X|Xs]) ->
    [ nocap(X) | nocaps(Xs) ].

nocap(X) ->
    case $A =< X andalso X =< $Z of
        true ->
            X+32;
        false ->
```

```
-:---  where3.erl    Top (14,39)    (Erlang EXT Flymake)
Wrote /Users/simonthompson/Desktop/where3.erl
```