Erlang I Homework 4

February 28, 2012

This section has 53 points.

1. (13 points) What is the result of the following entered into the erlang shell?
   (a) (1 point) `11<ten.`

   Solution: `true`

   (b) (1 point) `{123, 345} < [].`

   Solution: `true`

   (c) (1 point) `[boo, hoo] < [adder, zebra, bee].`

   Solution: `false`

   (d) (1 point) `[boo, hoo] < [boo, hoo, adder, zebra, bee].`

   Solution: `true`

   (e) (1 point) `{boo,hoo} < {adder,zebra,bee}.`

   Solution: `true`

   (f) (1 point) `{boo, hoo} < {boo, hoo, adder, zebra, bee}.`

   Solution: `true`

   (g) (1 point) `1.0 == 1.`

Solution: `true`

(h) (1 point) `1.0 =:= 1.`

Solution: `false`

(i) (1 point) `{1,2} < [1,2].`

Solution: `true`

(j) (1 point) `1 =< 1.2.`

Solution: `true`

(k) (1 point) `1 =/= 1.0.`

Solution: `true`

(l) (1 point) `(1<2) < 3.`

Solution: `false`

(m) (1 point) `(1 > 2) == false.`

Solution: `true`

2. (4 points) Which of the following are variable names, atoms or neither?

    1. `A_long_variable_name`
    2. `Flag`
    3. `january`
    4. `Name2`
    5. `fooBar`
    6. `DbgFlag`
    7. `node@ramone`
    8. `Node@Ramone`
    9. `Double`

10. `NewDouble`
11. `alfa21`
12. `Happy_days2`
13. `happy.days2`
14. `Happy.Days2`
15. `starts_with_lower_case`

Solution: Variables:

1. `A_long_variable_name`
2. `Flag`
3. `Name2`
4. `DbgFlag`
5. `Double`
6. `NewDouble`
7. `Happy_days2`

Atoms:

1. `january`
2. `fooBar`
3. `node@ramone`
4. `alfa21`
5. `happy.days2`
6. `starts_with_lower_case`

Neither:

1. `Node@Ramone`
2. `Happy.Days2`

3. (8 points) Create a data structure to store information about people. One is Joe Armstrong, shoe size 42 with two cats - zorro and daisy - and two children - Thomas (21) and Claire (17). The other is Mike WIlliams, shoe size 41 who likes boats and wine. Then create a structure to store these two people.

Solution:

```
JoeAttributeList = [shoeSize, 42, pets, [cat, zorro, cat, daisy],
children, [thomas,21,claire,17]].

JoeTuple = person, ‘‘Joe’’, ‘‘Armstrong’’, JoeAttributeList.

MikeAttributeList = [showSize, 42,likes,[boats,wine]].

MikeTuple = person, ‘‘Mike’’, ‘‘Williams’’, MikeAttributeList.

People = [JoeTuple, MikeTuple].
```

4. (3 points) Consider the following module:

```
-module(demo).
-export([double/1]).

% This is a comment.

double(Value) ->
  times(Value, 2).
times(X,Y) ->
  X*Y.
```

(a) (1 point) How would you compile this?

Solution: In the erlang shell type `c(demo)`.

(b) (1 point) What happens when you call `demo:times(3,5).` ?

Solution: `15`

(c) (1 point) What happens when you call `double(6).` ?

Solution: An error since the double function cannot be found.

5. (10 points) Write a module `shapes` that contains one function - `area`. This area function should work on squares, circle, triangles and returns an error for other shapes. If the three lengths of a triangle you may want to use the formula area $= S \cdot (S - A) \cdot (S - B) \cdot (S - C)$ where A, B, C are the length of the three sides and $S = \sqrt{(A + B + C)/2}$. Be sure to compile this function and test that it works.

Solution: shapes.erl:

```
-module(shapes).
-export([area/1]).

area({circle, Radius}) ->
  math:pi()*Radius*Radius;
area({square, Side}) ->
  Side*Side;
area({triangle, A, B, C}) ->
  S = (A+B+C)/2,
  math:sqrt(S*(S-A)*(S-B)*(S-C));
area(_) ->
  {error, "Unknown Shape"}.
```

To compile use `c(shapes).`

test_shapes.erl:

```
shapes:area({circle, 1}).
shapes:area({triangle, 1, 1, math:sqrt(2)}).
shapes:area({square, 2}).
shapes:area({rectangle, 4, 2}).
```

Running these tests get:

```
bash-3.2$ erl < tests_shapes.erl
Eshell V5.9  (abort with ^G)
1> 3.141592653589793
2> 0.49999999999999983
3> 4
4> {error,"Unknown Shape"}
5> *** Terminating erlang (nonode@nohost)
```

6. (15 points) Write a module `boolean.erl` that takes logical expressions and Boolean values (represented as the atoms `true` and `false`) and returns their Boolean results. The functions you write should include `b_not/1, b_and/2,` `b_or/2`, and `b_and/2`. You should not use the logical constructs `and, or` but instead use pattern matching to achieve your goal. Be sure to test your module. For example:

   $bool : b\_not(false) \rightarrow true$
   $bool : b\_and(false, true) \rightarrow false$
   $bool : b\_and(bool : b\_not(bool : b\_and(true, false)), true) \rightarrow true$

   Hint: implement `b_nand/2` using `b_not/1` and `b_and/2`.

Solution:

```
bash-3.2$ cat bool.erl
-module(bool).
-export([b_not/1, b_and/2, b_or/2, b_nand/2]).

b_not(true) ->
    false;
b_not(false) ->
    true;
b_not(Other) ->
    {error, "Must evaluate to atoms true or false"}.

b_and(true, true) ->
    true;
b_and(true, false) ->
    false;
b_and(false, true) ->
    false;
b_and(false, false) ->
    false;
b_and(Other, Other2) ->
    {error, "two arguments must be boolean atoms"}.

b_or(true, true) ->
    true;
b_or(true, false) ->
```

```
    true;
b_or(false, true) ->
    true;
b_or(false, false) ->
    false;
b_or(Other, Other2) ->
    {error, "two arguments must be boolean atoms"}.

b_nand(X, Y) ->
    b_not(b_and(X, Y)).

bash-3.2$ cat test_boolean.erl
bool:b_not(false).
bool:b_and(false, true).
bool:b_and(true, bool:b_not(bool:b_and(true, false))).
bool:b_nand(false, false).
bool:b_nand(true, true).
bool:b_or(bool:b_or(false, false), true).

bash-3.2$ erl < test_boolean.erl
Eshell V5.9  (abort with ^G)
1> true
2> false
3> true
4> true
5> false
6> true
7> *** Terminating erlang (nonode@nohost)
```