

Assignment Project Exam Help

Concurrent Programming

CS511

<https://powcoder.com>

Add WeChat powcoder

(Lack of) Types

Assignment Project Exam Help

Documenting Types using spec

Tail Recursion

<https://powcoder.com>

Exceptions

Add WeChat powcoder

Control Structures

Erlang is Strongly Typed

Assignment Project Exam Help

```
1 1> 6+"1".  
2 ** exception error: an error occurred when evaluating an  
3 arithmetic expression  
4   in operator +/2  
5   called as 6 + "1"
```

<https://powcoder.com>

Good, but there is no static type-checking...

Add WeChat powcoder

Recall from Previous Class

Assignment Project Exam Help

```
1 drivers_license(Age) when Age < 16 ->
2     forbidden ;
3 drivers_license(Age) when Age == 16 ->
4     'learners permit' ;
5 drivers_license(Age) when Age == 17 ->
6     'probationary license' ;
7 drivers_license(Age) when Age >= 65 ->
8     'vision test recommended but not required' ;
9 drivers_license(_) ->
10    'full license' ;
```

<https://powcoder.com>

Add WeChat powcoder

```
1 2> c1:drivers_license(45).  
2 'full_license'  
3 3> c1:drivers_license("hi").  
4 'vision test recommended but not required'
```

- ▶ What is going on?
- ▶ Recall the comparison order

number < *atom* < *reference* < *fun* < *port* < *pid* < *tuple* <
map < *nil* < *list* < *bitstring*

```
1 ...  
2 drivers_license(Age) when Age >= 65 ->  
3 'vision test recommended but not required' ;  
4 ...
```

Add WeChat powcoder

```
1 drivers_license(Age) when not(is_number(Age)) ->  
2   throw(wrong_argument_type);  
3 drivers_license(Age) when Age < 16 ->  
4   forbidden;  
5 % the rest follows without change
```

- ▶ <https://powcoder.com>
Other type checking predicates
is_atom/1, is_function/1, is_boolean/1, is_record/1,...

- ▶ More on exceptions later

```
1 9> c1:drivers_license(1.1)  
2 ** exception throw: wrong_argument_type  
3 in function c1:drivers_license/1 (c1.erl, line 6)
```

Add WeChat powcoder

(Lack of) Types

Assignment Project Exam Help

Documenting Types using spec

Tail Recursion

<https://powcoder.com>

Exceptions

Add WeChat powcoder

Control Structures

Assignment Project Exam Help

- ▶ Type specifications:

```
spec Function(ArgType1, ..., ArgTypeN) -> ReturnType.
```

or

```
spec Function(ArgName1 :: Type1, ..., ArgNameN :: TypeN) -> RT
```

- ▶ Type specifiers are used for:

- ▶ Documentation of intended usage
- ▶ Automatic detection of type errors

- ▶ The compiler does not check type but there are tools for doing this

<https://powcoder.com>

Add WeChat powcoder

Type Declarations – Examples

Assignment Project Exam Help

```
1 spec drivers_license(integer)) -> atom()  
2  
3 drivers_license(Age) when Age < 16 ->  
4     forbidden ;  
5 drivers_license(Age) when Age == 16 ->  
6     'earlier permit' ;  
7 drivers_license(Age) when Age == 17 ->  
8     'probationary license' ;  
9 drivers_license(Age) when Age >= 65 ->  
10    'vision test recommended but not required' ;  
11 drivers_license(_) ->  
12    'full license' .
```

<https://powcoder.com>

Add WeChat powcoder

Assignment Project Exam Help

- ▶ Checks that given specifications agree with call patterns
 - ▶ Also detects exceptions and dead code

- ▶ It does so loosely using so called “Success Typings”

http://www.it.u.se/research/group/hipe/papers/succ_types.pdf

- ▶ Assume that all is good in terms of typing (start from most general possible type) and then refining this view as the code analysis progresses

Add WeChat powcoder

Dialyzer

- ▶ Before using this tool you must initialize its internal tables (Persistent Lookup Tables)
- ▶ This process can take 5 minutes or more

```
1 $ dialyzer --eval_plt --apps erl kernel stdlib crypt
   mnesia sasl common_test eunit
2   Creating PLT /Users/ebonelli/.dialyzer_plt ...
3   Unknown functions:
4     compile:file/2
5     compile:forms/2
6     compile:noop_form/2
7     compile:output_generated/1
8     cover:analyse/2
9     cover:analyse_to_file/2
10    cover:analyse_to_file/3
11    cover:compile_beam/1
12    cover:export/1
13    cover:get_main_node/0
14    cover:import/1
15    cover:imported_modules/0
16    cover:start/0
17    cover:start/1
18    cover:stop/0
19    cover:stop/1
20    cover:which_nodes/0
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Checking Type Declarations

```
1 -spec drivers_license(integer()) -> atom().
2
3 drivers_license(Age) when Age < 16 ->
4   fo:children
5 drivers_license(Age) when Age == 16 ->
6   'learners permit' ;
7 drivers_license(Age) when Age == 17 ->
8   'probationary license' ;
9 drivers_license(Age) when Age >= 18 ->
10  'vision test recommended but not required' ;
11 drivers_license(_) ->
12  'full license'.
```

We check our code with dialyzer

```
1 $ dialyzer c1.erl
2   Checking whether the PLT /Users/ebonelli/.dialyzer_plt is
   up-to-date... yes
3   Proceeding with analysis... done in 0m1.03s
4 done (passed successfully)
```

Checking Type Declarations

```
1 -spec drivers_license(integer()) -> string().  
2  
3 drivers_license(Age) when Age < 16 ->  
4   forbidden;  
5 %...other clauses here...
```

We check our code with dialyzer

```
1 $ dialyzer c1.erl  
2   Checking whether the PLT /Users/ebonelli/.dialyzer_plt is  
   up-to-date... yes  
3   Proceeding with analysis...  
4 c1.erl:5: Invalid type specification for function c1:  
   drivers_license/1. The success typing is (_) -> '  
   forbidden' | 'full license' | 'learners permit' |  
   'probationary license' | 'vision test recommended but not  
   required'  
5   done in 0m1.09s  
6   done (warnings were emitted)
```

Checking Type Declarations

```
1 -spec drivers_license(integer()) -> string().
2
3 drivers_license(Age) when Age < 16 ->
4   fo:children
5 drivers_license(Age) when Age == 16 ->
6   'learners permit' ;
7 drivers_license(Age) when Age == 17 ->
8   "probationary license" ;
9 drivers_license(Age) when Age >= 18 ->
10  'vision test recommended but not required' ;
11 drivers_license(_) ->
12  'full license'.
```

We check our code with dialyzer

```
1 $ dialyzer c1.erl
2   Checking whether the PLT /Users/ebonelli/.dialyzer_plt is
   up-to-date... yes
3   Proceeding with analysis... done in 0m0.99s
4 done (passed successfully)
```

Assignment Project Exam Help

- ▶ Type variables can be used in specifications to specify relations for the input and output arguments of a function
- ▶ For example, the following specification defines the type of a polymorphic identity function:

```
-spec id(X) -> X.
```
- ▶ Notice that the above specification does not restrict the input and output type in any way

<https://powcoder.com>

Add WeChat powcoder

Type Declarations – More Examples

- ▶ Type variables can be constrained using a when clause
- ▶ The `::` constraint should be read as “is a subtype of”

```
1 %% sum(L) returns the sum of the elements in L
2 -spec sum(List) -> number() when
3     List :: [number()].
4
5 %% min(L) returns the minimum element of the list L
6 -spec min(List) -> Min when
7     List :: [T,...],
8     Min :: T,
9     T :: term().
10
11 %% append(X,Y) appends lists X and Y
12 -spec append(List1, List2) -> List3 when
13     List1 :: [T],
14     List2 :: [T],
15     List3 :: [T],
16     T :: term().
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

- ▶ Singletons can be either integers or atoms:

- ▶ 1, 2 or 42
- ▶ 'foo', 'bar' or 'atom'
- ▶ foo, 42

- ▶ Unions of singletons, what we normally refer to as “types”:

- ▶ integer(): any integer value
- ▶ float(): any floating point value
- ▶ atom(): any atom
- ▶ pid(): a process identifier
- ▶ ref(): a reference
- ▶ fun(): a function
- ▶ ... and many more

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Assignment Project Exam Help

- ▶ Type: for compound data structures
 - ▶ tuple(): a tuple of any form
 - ▶ list(): a proper list of any length

- ▶ Union type constructor

- ▶ type | type

```
1 -spec f('a' | 1) -> 'b' | 1.  
2 f(1) ->  
3   1;  
4 f(a) ->  
5   b.
```

<https://powcoder.com>

Add WeChat powcoder

Some built-in types and how they are defined¹:

term()	any()
boolean()	'false' — 'true'
byte()	0..255
char()	0..16#10ffff
nil()	[]
number()	integer() — float()
list()	[any()]
nonempty_list()	nonempty_list(any())
string()	[char()]
nonempty_string()	[char(),...]
function()	fun()
module()	atom()
no_return()	none()

¹http://erlang.org/doc/reference_manual/typespec.html

Defining Types – An Example

▶ Use of type directive

```
1  %%% {empty}    -- Empty tree
2  %%% {node,Data, LeftTree, RightTree}  -- Non empty tree
3
4  -type btree() :: {empty} | {node term(), btree() btree()}.
5
6  -spec sizeT(btree()) -> number().
7
8  sizeT({empty}) ->
9      0;
10 sizeT({node _,LT,RT}) ->
11     1 + sizeT(LT) + sizeT(RT).
```

Defining Types – Another Example

Assignment Project Exam Help

We would like to define our own type that specifies what a card looks like.

```
1 -type value() :: 1..13.  
2 -type suit() :: spade | heart | diamond | clubs.  
3 -type card() :: {card, suit(), value()}.  
4 -spec suit(card()) = suit().
```

Define the type of a deck of cards.

```
1 -type deck() :: list(card())
```

Add WeChat powcoder

An Example

```
1 -module(cards).
2 -export([kind/1, main/0]).
3
4 -type suit() :: spades | clubs | hearts | diamonds.
5 -type value() :: 1..10 | j | q | k.
6 -type card() :: {suit(), value()}.
7
8 kind(_, A) when A >= 1, A <= 10 -> number;
9 kind(_, _) -> error.
10
11 main() ->
12 number = kind({spades, 7}),
13 face = kind({hearts, k}),
14 number = kind({robies, j}),
15 face = kind({clubs, q}).
```

```
1 1> c1:main().
2 face
```

Somewhat unexpected...

An Example

Assignment Project Exam Help

```
1 $ dialyzer c1.erl
2   Checking whether the PLT /Users/ebonelli/.dialyzer_plt is
   up-to-date... yes
3   Proceeding with analysis...
4   done in 0m1.33s
5 done (warnings were emitted)
```

According to Dialyzer, everything is ok.

Add WeChat powcoder

An Example

```
1 -module(cards).  
2 -export([kind/1, main/0]).  
3  
4 -type suit() :: spades | clubs | hearts | diamonds.  
5 -type value() :: 1..10 | j | q | k.  
6 -type card() :: {suit(), value()}.  
7  
8 -spec kind(card()) -> face | number.  
9 kind({_, A}) when A == 1, A =< 10 -> number;  
10 kind(_) -> face.  
11  
12 main() ->  
13 number = kind({spades, 1}),  
14 face = kind({hearts, k}),  
15 number = kind({rubies, 4}),  
16 face = kind({clubs, q}).
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

An Example

Assignment Project Exam Help

```
1 $ dialyzer c1.erl
2   Checking whether the PLT /Users/ebonelli/.dialyzer_plt is
   up-to-date... yes
3   Proceeding with analysis...
4 c1.erl 34: Function main/0 has no local return
5 c1.erl:37: The call :l:kind({'rubies',4}) breaks the
   contract (card()) -> 'face' | 'number'
6   done in 0m1.02s
7   done (warnings were emitted)
```

<https://powcoder.com>
Add WeChat powcoder

(Lack of) Types

Assignment Project Exam Help

Documenting Types using spec

Tail Recursion

<https://powcoder.com>

Exceptions

Add WeChat powcoder

Control Structures

List Examples

Assignment Project Exam Help

```
1 > c(list_examples)
2 {ok,list_examples}
3 > list_examples:sum([1,2,3,4]).
4 10
5 > list_examples:len([0,1,0,1]).
6 4
7 > list_examples:append([5,4],[1,2,3]).
8 [5,4,1,2,3]
```

<https://powcoder.com>

- ▶ We will define them recursively (inductively)

▶ Base case: empty list (`[]`)

- ▶ Recursive case: a list with at least one element (`[x | xs]`)

Tail Recursion

- ▶ Programming pattern to increase performance
- ▶ It helps compilers when optimizing code
- ▶ Inefficient recursive definition

```
1 len([_:_:XS]) -> 1 + len(XS) ;  
2 len([]) -> 0.
```

Observe the evaluation of `len([1,2,3])`

```
1 len([1,2,3]) == 1 + len([2,3])  
2 len([1,2,3]) == 1 + 1 + len([3])  
3 len([1,2,3]) == 1 + (1 + (1 + len([]))) %%  
4 len([1,2,3]) == 1 + (1 + (1 + 0))  
5 len([1,2,3]) == 1 + (1 + 1)  
6 len([1,2,3]) == 1 + 2  
7 len([1,2,3]) == 3
```

- ▶ At the time of reaching the marked line, Erlang needs to keep in memory a long expression
- ▶ After that line, it starts shrinking the expression
- ▶ Imaging how it will work for a very big list!

Assignment Project Exam Help

- ▶ More efficiency by tail recursion
- ▶ Space (constant if we assume elements of the list have the same size)
- ▶ Efficiency (No returns from recursive calls)
- ▶ What is the trick?
 - ▶ Use of accumulators (partial results)
 - ▶ There are no more computations after the recursive call.

<https://powcoder.com>
Add WeChat powcoder

Tail Recursion

Assignment Project Exam Help

- ▶ We define `len_a`, the tail-recursive version of `len`.
- ▶ Function `len_a` has an extra parameters capturing the partial result of the function, i.e., how many elements `len_a` has seen so far

<https://powcoder.com>

```
1 len_a([_:_], Acc) -> len_a(XS, Acc+1),  
2 len_a([], Acc) -> Acc.
```

We define `len` based on `len_a` as follows

```
1 len(XS) -> len_a(XS, 0)
```

Add WeChat powcoder

What about the tail recursive version of `sum` and `append`?

(Lack of) Types

Assignment Project Exam Help

Documenting Types using spec

Tail Recursion

<https://powcoder.com>

Exceptions

Add WeChat powcoder

Control Structures

Assignment Project Exam Help

Three kinds:

- ▶ errors: run-time errors such as `1+a`; can be emulated with `error(Reason)`
- ▶ exits: generated error; generated by a process using `exit/1`
 - ▶ Studied next class
- ▶ throws: generated error; generated by a process using `throw/1`
 - ▶ Brief overview next

Add WeChat powcoder

Throw Exceptions

Assignment Project Exam Help

- ▶ Used for cases that the programmer can be expected to handle
- ▶ In comparison with exits and errors, they don't really carry any 'crash that process!' intent behind them, but rather control flow
- ▶ Good idea to document their use within a module using them

<https://powcoder.com>

```
1 1> throw(permission_denied)
2 ** exception thrown: permission_denied
```

Add WeChat powcoder

Try...Catch

```
1 -module(exceptions).  
2 -compile(export_all).  
3  
4 throws(F) ->  
5     try F() of  
6         _ -> ok  
7     catch  
8         _:throw -> {throw, caught, Throw}  
9     end.
```

```
1 1> c(exceptions).  
2 {ok, exceptions}  
3 2> exceptions:throws(fun() -> throw:throw, end).  
4 {throw, caught, thrown}  
5 3> exceptions:throws(fun() -> erlang:error(pang) end).  
6 ** exception error: pang
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Try..Catch

```
1 talk() -> "blah blah".
2
3 sword(1) -> throw(slice);
4 sword(2) -> erlang:error(cut_arm);
5 sword(3) -> exit(cut_leg);
6 sword(4) -> throw(punch);
7 sword(5) -> exit(cross_bridge).
8
9 black_knight(Attack) when is_function(Attack, 0) ->
10   try Attack() of
11     _ -> "None shall pass."
12   catch
13     throw(slice) -> "It is but a scratch.";
14     erlang:error(cut_arm) -> "I've had a core.";
15     exit:cut_leg -> "Come on you pansy!";
16     _:_ -> "Just a flesh wound."
17   end.
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Try-Catch

```
1 7> c(exception).
2 {ok,exceptions}
3 8> exceptions:talk().
4 "blah blah"
5 9> exceptions:black_knight(fun() -> exceptions:talk(0)).
6 "None shall pass."
7 10> exceptions:black_knight(fun() -> exceptions:sword(1) end
8 ).
9 "It is but a scratch."
10 11> exceptions:black_knight(fun() -> exceptions:sword(2) end
11 ).
12 "I've had worse."
13 12> exceptions:black_knight(fun() -> exceptions:sword(3) end
14 ).
15 "Come on you pansy!"
16 13> exceptions:black_knight(fun() -> exceptions:sword(4) end
17 ).
18 "Just a flesh wound."
19 14> exceptions:black_knight(fun() -> exceptions:sword(5) end
20 ).
21 "Just a flesh wound."
```

Assignment Project Exam Help

```
1 try Expr of
2   Pattern -> Expr1
3 catch
4   Type:Exception -> Expr2
5 after this always gets executed
6   Expr3
7 end
```

- ▶ Expr3 is always run, be there an exception or not

Add WeChat powcoder

Assignment Project Exam Help

```
1 1> catch throw(whya).
```

```
2 whya
```

```
3 2> catch exit(die).
```

```
4 {'EXIT',die}
```

```
5 3> catch 1/0.
```

```
6 {'EXIT',{badarith,[{erlang,'/',[1,0]}]}
```

```
7 {erl_eval,do_apply,5},
```

```
8 {erl_eval,expr,5},
```

```
9 {shell,exprs,6},
```

```
10 {shell,eval_exprs,6},
```

```
11 {shell,eval_loop,3}}}
```

```
12 4> catch 2/1.
```

```
13 4
```

<https://powcoder.com>

Add WeChat powcoder

(Lack of) Types

Assignment Project Exam Help

Documenting Types using spec

Tail Recursion

<https://powcoder.com>

Exceptions

Add WeChat powcoder

Control Structures

Assignment Project Exam Help

```
1 is_greater_than(X, Y) ->
2   if
3       X>Y ->
4       true;
5       true -> % works as an else branch
6       false
7   end
```

<https://powcoder.com>

Add WeChat powcoder

Assignment Project Exam Help

```
1 is_valid_signal(Signal) ->
2   case Signal of
3     {signal, _What, _From, _To} ->
4       true;
5     {signal, _What, _To} ->
6       true;
7     _Else ->
8       false
9   end.
```

<https://powcoder.com>

Add WeChat powcoder