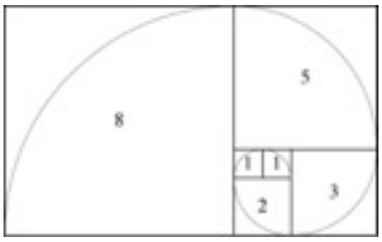
CO545 Spring Term 2013–14 Assessment 1 Solutions

Please make sure that you try to answer each of these questions: it's worth trying the next even if you can't manage the particular question that you are currently attempting.

Fibonacci function



[Image from hypnoticpeacock.com]

The Fibonacci sequence starts with `0` and `1` is generated by adding the previous two values to give the next:

```
0 1 1 2 3 5 8 13 21 34 55 ...
```

Define the function `fib` so that `fib(N)` is the `N`th Fibonacci number, for example:

```
fib(0) = 0
fib(3) = 2
fib(6) = 8 ...
```

Give an hand evaluation of `fib(4)`.

```
fib(0) -> 0;                            1 mark
fib(1) -> 1;
fib(N) -> fib(N-1) + fib(N-2).

fib(4)                                  1 mark
= fib(3)+fib(2)
= (fib(2)+fib(1))+(fib(1)+fib(0))
= ((fib(1)+fib(0))+fib(1))+(fib(1)+fib(0))
= ((1+0)+1)+(1+0)
= 3
```

Define the function `fib2` so that `fib2(N)` is the `N`th and `N+1`th Fibonacci numbers, paired together; for example:

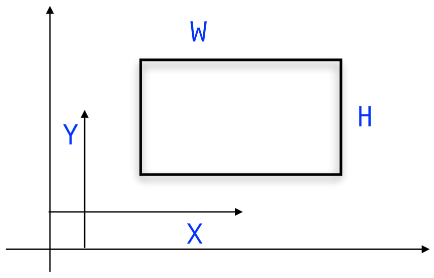
```
fib2(0) = {0,1}, fib2(5) = {5,8}
```

Give an hand evaluation of `fib2(4)` and explain how it differs from the evaluation of `fib(4)`.

```
fib2(0) -> {0,1};                      1 mark
fib2(N) ->
    {M,P} = fib2(N-1),
    {P,M+P}.
                                       1 mark
fib(4) = {P,M+P}
       |  {M,P} = fib2(3) = {P,M+P}
       |                  |  {M,P} = fib2(2) = {P,M+P}
       |                  |                  |  {M,P} = fib2(1) = {1,0+1}
       |                  |                  = {1,1+1}
       |                  = {2,1+2}
       = {3,2+3}
```

Big difference is that there's no repeated computation

Defining more Erlang functions: pattern matching



Based on the definition

```
area({circle, {X,Y}, R})
           -> math:pi()*R*R;
area({rectangle, {X,Y}, H, W})
           -> H*W.
```

Define a function that gives the *perimeter* of a circle and a rectangle, using a similar pattern-matching style.

```
perimeter({circle, {_,_}, R}) ->              2 marks, 1 per clause
   2*math:pi()*R;
perimeter({rectangle, {_,_}, H, W}) ->
   2*(H+W).
```

Challenge: Define a function which tests whether or not two shapes *overlap*. You might find it useful to use the function `abs` which gives the absolute value of a number, that is the number with the sign removed, so

```
abs(3)  = 3
abs(-9) = 9
```

Hint: first look at the cases of when two circles overlap and two rectangles overlap.

```
overlap({circle, {X1,Y1}, R1}, {circle, {X2,Y2}, R2}) ->
   math:sqrt((X1-X2)*(X1-X2)+(Y1-Y2)*(Y1-Y2)) =< (R1+R2);
overlap({rectangle,{X1,Y1},H1,W1}, {rectangle,{X2,Y2},H2,W2}) ->
   (max(X1+W1/2,X2+W2/2)-min(X1-W1/2,X2-W2/2)) =< W1+W2 andalso
   (max(Y1+H1/2,Y2+H2/2)-min(Y1-H1/2,Y2-H2/2)) =< H1+H2.
ADD FINAL CASE HERE                                   3 marks, 1 per clause
```

To check circle/rectangle. First check enclosing square and rectangle. If they don't cross, the answer is false. If they do intersect, need to check whether the circle and rectangle don't. This can only happen when one of the corners of the rectangle is in the square, but not in the circle.

Recursion over lists

Using recursion define these functions over lists of numbers:

- give the product of the numbers (and 1 in the case of an empty list), so that

```
     product([2,13]) = 26
```

```
product([])     -> 1;                    3 marks: 1 for base case,
product([X|Xs]) -> X*product(Xs).                 2 for recursion
```

- finds out how many times a given number appears in a list, so that

```
     occurs(13,[2,13,1,13]) = 2
     occurs(3,[2,13,1,13])  = 0
```

```
occurs(_,[])     -> 0;                   3 marks: 1 per clause
occurs(X,[X|Xs]) -> 1 + occurs(X,Xs);
occurs(X,[_|Xs]) -> occurs(X,Xs).
```

- selects only the even numbers in a list, so that

```
     evens([2,13,1,4,13]) = [2,4]
```

```
evens([])     -> [];                       3 marks: 1 per clause
evens([X|Xs]) when X rem 2 == 0 ->
   [X | evens(Xs)];
evens([_|Xs]) ->
   evens(Xs).
```

Rock-paper-scissors

Define a function that takes a list of outcomes of plays, as pairs of choices. and returns the overall score, from the point of view of the first player. For example:

```
     game_score([{paper,scissors},{paper,paper},{scissors,rock}]) = -2
```

```
game_score(Ps) ->
  lists:sum(lists:map(fun num_result/1, Ps)).   2 marks

                                               20 marks in TOTAL
```

Simon Thompson
18 February 2014