```
> module P1 where

[20pts] Consider the following text from page 169 of Mac Lane [1968]:

1:  [...] a function |z = f (x, y)| for all points |(x, y)| in some open
2:  set |U| of the cartesian |(x, y)|-plane.
3:  [...] If one holds |y| fixed, the quantity |z| remains just a
4:  function of |x|; its derivative, when it exists, is called the
5:  *partial derivative* with respect to |x|.
6:  Thus at a point |(x, y)| in |U| this derivative for |h ? 0| is

7:  ? z / ? x  =  f'_x (x, y)  =  \lim_{h \to 0} (f (x + h, y) - f (x, y)) / h

What are the types of the elements involved in the equation on the last line?
You are welcome to introduce functions and names to explain your reasoning.

----------------------------------------------------------------

Even though the exam question only asks for the last line we will here
analyse the text from the top.

Line 1: |z : Z|, |f : U -> Z|, |x : X|, |y : Y|, |(x, y) : (X, Y)|,

Line 2: |U : Powerset (X,Y)|, probably |X = Y = ?|, |Z = ?|

Line 3-4: For fixed |y|, the "quantity" |z| is a function of |x|. Lets
name this family of functions |g y x = f (x, y)|.

Line 5: For a fixed |y| the partial derivative of |f| with respect to
|x| is the "normal" derivative of the function |g y : X -> Z|. For all
|y| we can call this derivative |g' y = D (g y) : X -> Z|.

Line 6: Here we pick a fixed (but arbitrary) |(x, y)| and (implicitly)
introduce a value |h : H| with (presumably) |H = ? - {0}|.

Line 7 (the last line): From the context we know that this line gives
the value of the partial derivative at one specific point |(x, y)|.
Thus |? z / ? x| is implicitly applied to |(x, y)|. We could write
|(?z/?x)(x,y) : Z| and thus |(?z/?x) : U -> Z| if we generalise to a
function from arbitrary points. Note that |x| in |?x| is just a label
and not the value of |x| in scope from Line 6. We could say that the
operator |?/?x| has type |(U->Z) -> (U->Z)|

Similarly |f'_x (x,y) : Z|, |f'_x : U -> Z|, and the |x| here is again
just a text label. The post-fix operator "prime and subscript x" has
type |(U->Z) -> (U->Z)|.

Then the limit expression: First remember that |(x,y)| is fixed, so
the only "varying variable" here is |h|. If we rewrite to use

< lim : (H -> Z) -> {p | p ? ?, Limp p H} -> Z

(from the lecture notes Weeks 2-3) we get the expression

< lim (\h -> (f (x + h, y) - f (x, y)) / h) 0

or if we want to name the anonymous function we could say

> psi :: (U -> Z) -> U -> H -> Z
> psi f (x, y) h = (f (x + h, y) - f (x, y)) / h

The limit is then |lim (psi f (x, y)) 0|.

----------------

Just for type-checking:

> data X
> data Y
> type U = (X,Y)
> type Z = X
> type H = Z
> f :: U -> Z
> f = undefined
> instance Num X
> instance Fractional X
> main = undefined
```

```
----------------------------------------------------------------

> module P2 where
> import PS
> default (Rational, Integer)

[25pts] Consider the following differential equation:

    f'' t + 4*f t = 6*cos t,       f 0 = 0,   f' 0 = 0

    * [10pts] Solve the equation assuming that |f| can be
      expressed by a power series |fs|, that is, use |integ| and the
      differential equation to express the relation between |fs|, |fs'|,
      |fs''|, and |rhs| where |rhs| is the power series representation
      of |(6*).cos|.

    What are the first four coefficients of |fs|?

----------------

> fs    =  integ fs' 0
> fs'   =  integ fs'' 0
> fs''  =  rhs - 4*fs

One way of describing |cos| is as a solution to |g'' = -g|.

> rhs   =  integ rhs' 6
> rhs'  =  - (integ rhs 0)

Short: Hand-computing these series a few steps (more than asked for in
the exam question) gives us

> hfs    =  0:  0:   3:  0: (-5)/4:[]
> hfs'   =  0:  6:   0: -5: []
> hfs''  =  6:  0: -15: []

> hrhs   =  6:  0:  -3:  0: (1/4): []
> hrhs'  =  0: -6:   0:  1:  0   : []

Details of the method: start by filling in the constant term for uses of |integ|

< fs    =  0:
< fs'   =  0:
< fs''  =
<
< rhs   =  6:
< rhs'  =  0:

Then compute |head fs''| from |head rhs == 6| and |head fs == 0|

< fs''  =  6:

Now we fill in the second coefficients for uses of |integ|

< fs    =  0:0:
< fs'   =  0:6:
< fs''  =  6:
<
< rhs   =  6: 0:
< rhs'  =  0:-6:

In fact we can compute |fs| one step further as |(fs!!1) / 2|

< fs    =  0:0:3

Then we compute |fs''!!1| as before: from |rhs!!1 == 0| and |fs!!1 == 0|

< fs''  =  6:0

And this can be propagated to |fs'| and |fs|

< fs    =  0:0:3:0
< fs'   =  0:6:0

We have now reached the first four terms as requested: |f t ~= 3*t^2|

----------------
```

```
    * [15pts] Solve the equation using the Laplace transform.  You
        should need only two formulas (and linearity):

<   L (\t -> exp (a*t)) s  =  1 / (s - a)
<               2 * cos t  =  e^{i*t} + e^{-i*t}

Helper computations, using the initial conditions:
  L f'  s  =  -f 0  + s*L f s   ={here}=    s*L f s
  L f'' s  =  -f' 0 + s*L f' s  ={here}=  s^2*L f s

Start from the point free form of the equation:

  f'' + 4*f = (6*).cos

apply |\g -> L g s| to both sides to get LHS and RHS.

  LHS
=
  L (f'' + 4*f) s
= -- linearity
  L f'' s + 4*L f s
= -- helper computation of L f'' s
  (s^2 + 4) * L f s
=
  RHS
=
  L ((6*).cos) s
= -- the |cos| equation, Laplace of exponential, linearity
  3*(1/(s-i) + 1/(s+i))

Thus we have

  (s^2 + 4) * L f s  =  3*(1/(s-i) + 1/(s+i))

Factoring (and juxtaposition for multiplication) using
  s^2 + 4 = (s-2i)(s+2i)
Multiply by (s-i)(s+i) on both sides:
  (s-i)(s+i)(s-2i)(s+2i)L f s  =  3(s+i) + 3(s-i)  =  6s

Ansatz:
  L f s = A/(s-i) + B/(s+i) + C/(s-2i) + D/(s+2i)
We then get
  6s  =      (s+i)(s-2i)(s+2i)A
      + (s-i)     (s-2i)(s+2i)B
      + (s-i)(s+i)      (s+2i)C
      + (s-i)(s+i)(s-2i)      D

Let s=i:    6i = (i+i)  (i-2i) (i+2i)  A = (2i)(-i)(3i)  A = 6iA   => A = 1
Let s=-i:  -6i = (-i-i) (-i-2i)(-i+2i) B = (-2i)(-3i)(i) B = -6iB  => B = 1
Let s=2i:  12i = (2i-i) (2i+i) (2i+2i) C = (i)(3i)(4i)   C = -12iC => C = -1
Let s=-2i:-12i = (-2i-i)(-2i+i)(-2i-2i)D = (-3i)(-i)(-4i)D = 12iD  => D = -1

Thus
  L f s = (1/(s-i) + 1/(s+i)) - (1/(s-2i) + 1/(s+2i))
which we recognise as the transform of exponentials
  f t = (exp(it)+exp(-it)) - (exp(2it)+exp(-2it))
and we can apply the |cos| formula again
  f t = 2*cos t - 2*cos (2*t)

Last step: check the original specification:

First compute derivatives:
  f' t  = -2*sin t - 4*sin (2*t)
  f'' t = -2*cos t + 8*cos (2*t)

Then check boundary conditions:
  f 0  =  2*cos 0 - 2*cos 0 = 2 - 2 = 0   -- OK
  f' 0 = -2*sin 0 - 4*sin 0 = 0 - 0 = 0   -- OK

Finally check the diff. eq.:

  f'' t + 4*f t
=
  (-2*cos t + 8*cos (2*t)) + 4*(2*cos t - 2*cos (2*t))
=
  6*cos t

OK.
```

```
----------------------------------------------------------------

[20pts] One definition of *derivative* is (inspired by [Rudin, 1964], p. 89):

    *Definition:* Let |f : [a, b] -> ?|.
    For an |x ? [a, b]|, consider the function |phi f (x) : [a, b] -> ?| by

<   phi f (x) (t)  =  (f(t) - f(x))/(t - x),    -- for |t /= x|

    and define

<   f'(x)  =  lim_{t->x} phi f (x)(t)

    provided that this limit exists.  We thus associate with |f| a
    function |f'| whose domain of definition is the set of points |x|
    at which the limit exists; |f'| is called the *derivative* of |f|.


    * [5pts] Let |r : [1, 2] -> ?| with |r(x) = 1/x|.
      Compute |r'| using this definition.

< phi r x t = (r(t)-r(x))/(t-x) = (1/t-1/x)/(t-x) = (x-t)/x/t/(t-x) = -1/x/t

Note that |-1/x/t| is defined also for |x=t| (as long as x/=0). Thus
the limit computation is trivial: |lim (phi r x) t = 1/x/t|

Thus we have:

  r' x = lim (phi r x) x = lim (\t->-1/x/t) x = -1/x^2

as expected.

    * [5pts] Let |h = g . f| for |f, g : [a, b] -> [a, b]|.
      Formulate the chain rule (the derivative of |h| in terms of
      operations on |f| and |g|).

I write |D f| for the derivative of |f| for clarity.

The chain rule: for |h = g . f| we have |D h = (D g . f)*D f|.

    * [10pts] Prove your formulation of the chain rule using the
      definition above.

  D h x
= -- def. of derivative
  lim (phi h x) x
= -- def. of h
  lim (phi (g . f) x) x

Let us do a sub-computation (prove a lemma) for arbitrary |t|:

  phi (g . f) x t
= -- def. of |phi|
  ((g . f)(t) - (g . f)(x))/(t - x)
= -- def. |.|
  (g(f t) - g(f x))/(t - x)
= -- Assume |f t - f x /= 0|
  (g(f t) - g(f x))/(f t - f x) * (f t - f x)/(t - x)
= -- def. of |phi|
  phi g (f x) (f t) * phi f x t

Now we can continue our equality proof:

  lim (phi (g . f) x) x
= -- Our new lemma
  lim (\t -> phi g (f x) (f t) * phi f x t) x
= -- limit of a product of functions is the product of the limits
  lim (\t -> phi g (f x) (f t)) x * lim (phi f x) x
= -- limit of a composition (using cont. of |f|) + def. of D f
  lim (\ft -> phi g (f x) ft) (f x) * D f x
= -- eta-reduction
  lim (phi g (f x)) (f x) * D f x
= -- def. of |D g| (at the point |f x|)
  D g (f x) * D f x
= -- Def. of |.| and |*| on functions
  ((D g . f) * D f) x

Thus we get |D h = (D g . f) * D f|.
```

```
-----------------------------------------------------------------

> module P4 where

[15pts] Recall the type of expressions

\begin{code}
data FunExp  =  Const Rational
             |  Id
             |  FunExp  :+:  FunExp
             |  FunExp  :*:  FunExp
             |  FunExp  :/:  FunExp
             |  Exp  FunExp
             |  Sin  FunExp
             |  Cos  FunExp
             -- and so on
             deriving Show
\end{code}

and consider the function

\begin{code}
f :: Double -> Double
f x = exp (sin x) + x
\end{code}


* Find an expression |e| such that |eval e == f| and show this
  using equational reasoning.

Here is a calculational version of the solution:

<   f
< == (+) instance for functions
<    (exp . sin . id) + id
< == exp instance for functions
<   exp (sin . id) + id
< == sin instance for functions
<   exp (sin id) + id
< == eval for Id
<   exp (sin (eval Id)) + (eval Id)
< == eval for Sin
<   exp (eval (Sin Id)) + (eval Id)
< == eval for Exp
<   eval (Exp (Sin Id)) + (eval Id)
< == eval for (:+:)
<   eval (Exp (Sin Id) :+: Id)

Thus |e = Exp (Sin Id) :+: Id| gives |eval e == f|


* Implement a function |deriv2| such that, for any |f : Fractional
  a => a -> a| constructed with the grammar of |FunExp| and any |x| in
  the domain of |f|, we have that |deriv2 f x| computes the second
  derivative of |f| at |x|.
  Use the function |derive :: FunExp -> FunExp| from the lectures
  (|eval (derive e)| is the derivative of |eval e|).
  What instance declarations do you need?

  The type of |deriv2 f| should be |Fractional a => a -> a|.

\begin{code}
deriv2 f x = eval (derive (derive (f Id))) x
\end{code}

The instance declarations needed are:

\begin{code}
instance Num FunExp              -- ...
instance Fractional FunExp       -- ...
instance Floating FunExp         -- ...
instance Num a => Num (x -> a)  -- ...
-- etc.
\end{code}

-----------------------------------------------------------------
```

```
> {-# LANGUAGE FlexibleInstances #-}
> {-# LANGUAGE UndecidableInstances #-}
> module P5 where
```

[20pts] Consider a non-deterministic system with a transition function |f : G -> [G]| (for |G={0..5}|) represented in the following graph [elided]. The transition matrix can be given the type |m :: G -> (G -> Bool)| and the canonical vectors have type |e i :: G -> Bool| for |i| in |G|.

* (General questions.) What do the canonical vectors represent? What about non-canonical ones? What are the operations on |Bool| used in the matrix-vector multiplication?

* (Specific questions.) Write the transition matrix |m| of the system. Compute, using matrix-vector multiplication, the result of three steps of the system starting in state |2|.

a) Canonical vestors represent nodes or singleton subsets of |G|.
Non-canonical vectors represent sets of nodes (partial knowledge).

```
> instance Num Bool where
>   (+) = (||)
>   (*) = (&&)
>   fromInteger 0 = False
>   fromInteger 1 = True
```

b) The columns of the transition matrix show where you may get from a certain node.

```
m = |001000|
    |100000|
    |000010|
    |011010|
    |000000|
    |010100|
```

let tr = transpose of a vector

```
v0 = (2==) = tr (001000)
v1 = mult m v0 = tr (100100)
v2 = mult m v1 = tr (010001)
v3 = mult m v2 = tr (000101)
-- for fun:
v4 = mult m v3 = tr (000001)
v5 = mult m v4 = tr (000000)
```

and in general m^5 = 0