```
(* Let's start with : *)

Variable T:Type.

(* This declares that T is a type. *)

Variable P:T -> Type.

(* This declares that P is a family of types parametrized by T.
Geometrically we may think about it as being a fibration over T. *)

Variable t:T.

(* This declares a point of T. *)

Definition X1:= P t.

(* This defines X1 as the fiber of our family P over t. *)

Definition X2:= forall x:T, P x.

(* This is an example of one of the key constructions - the "dependent
product". It defines
X2 as the space of families of points in P parametrized by T i.e. the
space of sections of the corresponding bundle. *)

Variable T':Type.

(* Consider another type T'. *)

Definition constT':= fun x:T => T'.

(* Let constT' be the constant family over T corresponding to T'. *)

Definition X3:= forall x:T, constT' x.

(* Now X3 is the space of sections of this family i.e. the space of
functions from T to T'. One can also write X3 as T -> T'. *)



(* We now consider one of the key features of the language. In the
following line we define for any X and x:X a family "paths X x -" which
is "inductively defined" by the condition that it is given with a
```

```
section idpath over x itself. Inductively defined roughly means
"universal" among families with a section over x. Homotopy-
theoretically this corresponds to the decomposition of the map pt -> X
corresponding to x into a weak equivalence and a fibration. As we know
the fiber of such a fibration over x0:X will be a model for the paths
space paths X x x0. *)

Inductive paths (X:Type)(x:X): X -> Type := idpath: paths X x x.

(* Any inductive definition generates together with the inductive
object itself the induction rule associated with this object. In fact
in the current Coq system it generates three inductive rules for most
inductive definitions, but this is irrelevant for us. We need only the
most general one *_rect, in this case paths_rect. *)

Print paths_rect.

(* Printing paths_rect tells us that it has the following type:

paths_rect:  forall (X : Type) (x : X) (P : forall x0 : X, paths X x x0
-> Type),
       P x (idpath X x) -> forall (y : X) (p : paths X x y), P y p

The most remarkable fact that this is exactly the condition that for
each X and x the inclusion from the point into the "total space" [paths
X x *] of the bundle corresponding to the family x0 |-> paths X x x0 is
a trivial cofibration. I.e. that it satisfies the Quillen's left
lifting property for the fibrations (= families of types). Indeed, P of
the type forall x0 : X, paths X x x0 -> Type is a mapping which assigns
to any point (x0:X, e: paths X x x0) of the total space [paths X x *] a
type i.e. a fibration over this total space. paths_rect then gives a
mapping from P x (idpaths X x) which is the type of sections of this
fibration over pt to the type of sections of this fibration over the
whole total space. There is also the so called "iota-reduction rule"
associated with any inductive definition which in this case says
precisely that the section obtained in this way is an extension of the
original section over pt. *)

(* Here are some other inductive definitions: *)

Inductive emptytype :=.

(* - the empty type. *)

Inductive unit := tt: unit.
```

```
(* - the one point type. *)

Inductive nat := O:nat | S:nat -> nat.

(* - natural numbers. *)

Inductive total2 (X:Type)(Q:X -> Type) : Type := tpair: forall x:X,
forall y: Q x, total2 X Q.

(* This is a construction which associates to any family of types the
total space of the corresponding fibration. Its inductive form says
essentially that for any x:X and y: Q x one can associate in a
canonical way a point tpair X Q x y of the total space and that the
total space is universal among spaces for which such association is
available. *)


(* Let me now define the conditions of contractibility and of being of
levels 1 and 2 for types. *)

Definition iscontr (X:Type) := total2 X (fun x:X => forall y:X, paths X
x y).

(* This is the contractibility condition on X i.e. I calim that X is
contractible if and only if iscontr X has a point and that (iscontr X)
is a "truth value" i.e. is itself of level 1. Let us check these
properties on the intuitive level. Suppose iscontr X has a point (x,
f). Then X has a point x and in particular is non-empty. Let see what f
is. We have the fibration [paths X x *] over X and f is a section of
this fibration. But [paths X x *] is contractible for any X and if its
projection to X has a section then X must be contractible. Suppose now
that X is contractible. Then, I calim, iscontr X is contractible and in
particular has a point. This I leave as an excersise in basic homotopy
theory (hint: any fibration over a contractible space is trivial and
the space of sections of a fibration with contractible fibers is
contractible.). *)


Definition isoflevel1 (X:Type) := forall x1:X, forall x2:X, iscontr
(paths X x1 x2).

(* This is the condition of being of level 1 as defined in the first
part of the lecture. *)
```

```
Definition isaset (X:Type) := forall x1:X, forall x2:X, isoflevel1
(paths X x1 x2).

(* This is the condition of being of level 2 i.e. of being a set. *)

Theorem th1: isaset nat.
Proof. auto. unfold isaset. intros. Admitted.

(* This theorem asserts that natural numbers form a set. Actually at
the moment I do not have a proof of it since I was concentrating on
more elementary results such as the main properties of weak
equivalences. I am pretty sure however that it is provable - to appear
in one of the next udates of the 'foundations" files which can be found
on my web page. *)
```