Chapter 1

Library coq

COQ en COQ `Require Import` *String List PeanoNat.*

`Open Scope` *string_scope.*
`Open Scope` *list_scope.*
`Import` *ListNotations.*

définition du λ-terme `Inductive` *terme* : `Set` :=
| *V* : *string* → *terme*
| *C* : *string* → *terme*
| *App* : *terme* → *terme* → *terme*
| *Lam* : *string* → *terme* → *terme* → *terme* .

`Definition` *t* : *terme* := *Lam* "x" (*C* "type") (*V* "x").
`Definition` *env* : *list* (*string*×*string*) := ("a","1") :: ("b","2") :: ("c","3") :: *nil* .

`Fixpoint` *mem* (*x*:*string*) (*l*:*list string*) : *bool* :=
`match` *l* `with`
| *nil* ⇒ *false*
| *h*::*t* ⇒ `if` *h*=?*x* `then` *true* `else` *mem x t*
`end`.

`Check` *hd.*
`Print` *hd.*

`Compute` (*hd* (1::2::3::*nil*))
.

`Fixpoint` *union* (*l1 l2*: *list string*) : *list string* :=
`match` *l1* `with`
| *a1*::*r1* ⇒ `if` *mem a1 l2* `then` *union r1 l2*
`else` *a1* :: (*union r1 l2*)
| *nil* ⇒ *l2*
`end`.

`Fixpoint` *remove* (*var*:*string*) (*l*: *list string*) : *list string* :=
`match` *l* `with`

```
  | h::t ⇒ if h =? var then remove var t
            else h::(remove var t)
  | nil ⇒ nil
  end.

Fixpoint varLibres (lambdaTerm:terme) : list string :=
  match lambdaTerm with
  | V x ⇒ [ x ]
  | C _ ⇒ []
  | App n m ⇒ union (varLibres n) (varLibres m)
  | Lam x tx m ⇒ union (remove x (varLibres m)) (remove x (varLibres tx))
  end.

Compute (remove "a" ["b"; "c"; "a"]).

Compute union ("a"::"b"::"c"::nil) ("a"::"z"::"q"::nil) .

Check [1;2;3].
```