Difficultés :  
  
-> Comprendre, apprendre et utiliser les fonctions associées au lists !  
Utilisées lors de ce Seminar :

* lists :keydelete
* lists :keyfind
* lists :flatmap
* lists :foldl
* lists :sort
* lists :keyreplace
* lists :map
* lists :member
* lists :foreach

(Mais on a aussi du utiliser dictionnary ! avec

* dict :new
* dict :append
* dict :find
* dict :erase

)

-> Le temps ! Putain c’est super long à faire et avec les autres matières que j’ai c’est compliqué d’être autant investit dans un seminar  
-> Beaucoup plus compliqué que le seminar sur le petit serveur web, difficulté non croissante

module map :  
Ça commence avec le premier module dans lequel on doit définir des fonctions associées à notre map dont on va se servir tout au long du TP. Des fonctions de bases sur les map telles que retourner toutes les nodes de la map ou mettre a jour la map, seulement lorsqu’il s’agit de coder le tout en Erlang tout devient plus compliquer quand on ne connaît pas encore bien le langage. Après beaucoup de recherche et de lecture sur le fonctionnement des lists et des fonctions citées ci-dessus on parvient tout de même à obtenir un module fonctionnel.   
  
module(dijkstra) :

Pour ce qui est de l’implémentation de l’aglorithme de Dijkstra qui, on le rappelle, permet d’obtenir dans un graphe la distance la plus courte à partir d’un point pour aller vers un autre. Cet algorithme est beaucoup utilisé dans les problèmes liés au routing car il permet en effet de savoir quel est le coût le plus faible pour aller d’une destination à un autre à partir d’un certain router. Pour notre router il nous a donc fallut implémenter cet algorithme en Erlang, et ça n’a pas été une mince à faire. Sans l’aide d’internet et de beaucoup de recherches je n’aurais personnelement pas réussi je pense à implémenter toutes les fonctions du modules seulement avec erlang.doc. Il faut pour implémenter Dijkstra comprendre parfaitement le fonction des fonctions telles que lists :leyfind, litsts :keyreplace ou encore lists :foldl. C’est ce qui m’a posé le plus de problème pour cette étape. Je comprends très bien le fonction de Dijkstra pour l’avoir étudier dans un cours d’algorithmie sur la theorie des graphes, je comprends aussi très bien l’utilité de certaines fonctions du module lists pour se faire, mais il a été très compliqué pour moi d’arriver à implémenter chacunes des fonctions demandées mais après beaucoup de temps je pense que mon module dijkstra est fonctionnel. C’est ce que les tests laissent sous entendre en tout cas. Et c’est aussi grâce aux résultats de tests donnés dans le sujet que j’ai pu comprendre comme effectuer certaines fonctions.

Module(interfaces) :

En effet comme le dit le sujet une fois que l’on a implémenté le module Dijkstra, implémenter le module interface semble beaucoup plus rapide.   
Il s’agit ici d’ajouter à notre router la capacité d’avoir des interfaces (prendre la phrase d’intro de la partie)  
  
module(history) :  
  
J’ai utilisé le module dict car après avoir chercher sur internet comment faire une sorte de hashing en erlang il semblerait que ce module soit efficace.

Because Dictionaries are data structures that provide a mapping between keys and values. Like lists, they are another way of storing variable amounts of information, but differ in that you can retrieve random elements in constant time.

Module(router) :

Erreur dans le sujet il ne faut pas faire router(Name, 0, Msgs, Intf, Table, Map) mais router(Name, 0, Hist, Intf, Table, Map).

Sinon tout est donné mais je n’ai pas fait la fonction qui affiche le truc, par manque de temps. Je voulais utiliser le temps qu’il me restait pour faire des tests.

Testing what we have :  
Macintosh-4:Seminars\_3 pierre$ erl -name sweden@192.168.1.13 -setcookie routy -connect\_all false

Erlang/OTP 18 [erts-7.0] [source] [64-bit] [smp:4:4] [async-threads:10] [hipe] [kernel-poll:false]

Eshell V7.0 (abort with ^G)

(sweden@192.168.1.13)1> c(routy).

routy.erl:62: Warning: variable 'From' is unused

{ok,routy}

(sweden@192.168.1.13)2> c(dijkstra).

dijkstra.erl:57: Warning: variable 'Dest' is unused

{ok,dijkstra}

(sweden@192.168.1.13)3> c(hist).

{ok,hist}

(sweden@192.168.1.13)4> c(map).

{ok,map}

(sweden@192.168.1.13)5> c(interfaces).

{ok,interfaces}

(sweden@192.168.1.13)6> routy:start(r1,stockholm).

true

(sweden@192.168.1.13)7> routy:start(r2,lund).

true

(sweden@192.168.1.13)8> lund ! {add,stockholm,{stockholm,'sweden@192.160.1.13'}}.

\*\* exception error: bad argument

in operator !/2

called as lund ! {add,stockholm,{r1,'sweden@192.160.1.13'}}

(sweden@192.168.1.13)9>

Je n’arrive pas à trouver ni comprendre cette erreur… Donc je ne peux pas effectuer de tests sur mon routeur, sur lequel j’ai passé pourtant beaucoup (trop ?) de temps. J’ai verifié si il y avait des erreurs pour chacuns de mes modules il ne semble pas y en avoir, j’ai vérifié si les modules était bien loaded (m().) c’est le cas. Donc je ne sais pas. Il semble me dire que le deuxième argument que j’envoie est mauvais pourtant je ne fais que recopier ce qu’il y a écrit dans le sujet ? Impossible de trouver après 3h dessus. J’abandonne.   
  
Conclusion : Très dur seminar à mon goût mais très interessant, on voit comment implémenter le fonction de l’algorithme dijkstra dont on nous parle tant dans les cours d’internetworking ou autre, on utilise profondément les procedures associées au module list. C’est un très long et dur seminar a mes yeux car au final lors du moment de tout tester j’ai une erreur que je ne comprends pas et que je n’arrive pas à régler ce qui laisse à un goût amer à cette expérience.