Corrigé. T.D.1

I.1. Il s'agit de tester si les indices i et j sont bien compris entre 0 et 7 :

```
let valide (i,j)= i>=0 && i<=7 && j>=0 && j<=7;;
```

I.2. On insère dans une liste l initialement vide les cases valides que l'on peut atteindre depuis la position (i, j) :

```
let coupSuivant (i,j)=
   let ajoute (a,b) = (a+i,b+j) in
   let deplacement=[|(1,2);(-1,2);(1,-2);(-1,-2);(2,1);(-2,1);(2,-1);(-2,-1)|] and l = ref [] in
      for k=0 to 7 do
         let c=ajoute deplacement.(k) in
            if valide c then
               l := c::(!l)
      done;
      !l;;
```

I.3. On peut remarquer que les cases de l'échiquier qui peuvent être atteintes à partir de la case (i_0, j_0) en un nombre minimum de coups égal à $k+1$ font partie de celles obtenues en appliquant `coupSuivant` aux cases pouvant être atteintes en un nombre minimum de coup égal à k. On va donc remplir la matrice M en y mettant d'abord l'unique 0 (correspondant à la case (i_0, j_0)), puis les 1, puis les 2 etc. D'autre part, on peut remarquer qu'un déplacement en un nombre minimum de coups entre deux cases passe nécessairement par des cases 2 à 2 distinctes donc comporte au maximum 63 coups.

Nous procédons de la manière suivante :

- nous initialisons la matrice M : la case (i_0, j_0) contient la valeur 0 et toutes les autres cases contiennent la valeur $+\infty$ (pratiquement, on représentera $+\infty$ par la valeur 64 d'après la remarque précédente) ;
- nous initialisons une référence i à la valeur 0 et une liste L à la valeur $[(i_0, j_0)]$: à chaque étape du calcul, L est la liste des cases que l'on peut atteindre depuis (i_0, j_0) en un nombre minimal de coups égal à i ;
- tant que L est non vide, on crée une liste LL contenant tous les successeurs des couples (a, b) stockés dans L et non encore rencontrés : on incrémente i, on affecte aux entrées de M associées à ces nouvelles cases la valeur (nouvelle) de i et on remplace L par LL.

Voici les deux premières étapes du fonctionnement de l'algorithme, quand $i_0 = j_0 = 0$:

- Initialisation : on définit M matrice 8×8 contenant la valeur 64 et on affecte à $M[0, 0]$ la valeur 0 ; on affecte à i la valeur 0 et à L la valeur $[(0, 0)]$.
- Première étape : les successeurs de $(0, 0)$ non encore rencontrés sont les éléments de la liste $LL = [(1, 2); (2, 1)]$: i prend la valeur 1, la matrice M devient (en notant ∞ au lieu de 64, pour faciliter la

lecture) :

$$M = \begin{pmatrix} 0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 1 & \infty & \infty & \infty & \infty & \infty \\ \infty & 1 & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \end{pmatrix}$$

et L devient $[(1,2);(2,1)]$.

- Deuxième étape : les successeurs de $(2,1)$ ou $(1,2)$ non encore rencontrés sont les éléments de la liste $LL = [(4,0);(0,2);(4,2);(1,3);(3,3);(3,1);(2,0);(0,4);(2,4)]$: i prend la valeur 2, la matrice M devient

$$M = \begin{pmatrix} 0 & \infty & 2 & \infty & 2 & \infty & \infty & \infty \\ \infty & \infty & 1 & 2 & \infty & \infty & \infty & \infty \\ 2 & 1 & \infty & \infty & 2 & \infty & \infty & \infty \\ \infty & 2 & \infty & 2 & \infty & \infty & \infty & \infty \\ 2 & \infty & 2 & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \end{pmatrix}$$

et L devient $[(4,0);(0,2);(4,2);(1,3);(3,3);(3,1);(2,0);(0,4);(2,4)]$.

Le calcul de LL se fait facilement : on initialise une liste LL à la valeur [], on lit l'un après l'autre chaque couple (a,b) stocké dans L, et pour chaque successeur (c,d) de (a,b) non encore rencontré, on modifie l'entrée (c,d) de M et on insère (c,d) dans LL. Le parcours des listes se fera bien évidemment au moyen de références :

```
let cavalier (i0,j0)=
   let m=Array.make_matrix 8 8 64 and i=ref 0 and l=ref [(i0,j0)] in
      m.(i0).(j0) <- 0;
      while !l<>[] do         (* tant qu'il y a des cases à étudier *)
         incr i;
         let ll = ref [] in                  (* on initialise ll *)
            while !l<>[] do                  (* on parcourt l *)
               let (a,b)=List.hd(!l) in      (* pour chaque élément (a,b) de l *)
                  l := List.tl(!l);
                  let pile=ref (coupSuivant (a,b)) in
                  while !pile<>[] do            (* on étudie chaque successeur de (a,b) *)
                     let (c,d)=List.hd(!pile) in
                     pile := List.tl(!pile);
                     if m.(c).(d)=64 then   (* si le successeur (c,d) est nouveau *)
                        begin
                        m.(c).(d) <- !i;           (* il est à la distance i de (i0,j0) *)
                        ll := (c,d)::(!ll)         (* et on l'ajoute à ll *)
                        end
                  done;
            done;                                  (* l a été vidé *)
            l := !ll;                              (* l prend sa nouvelle valeur *)
      done;
      m;;
```