Module Tri

1. Tri fusion (*mergesort*). Une première fonction *decoupe* découpe une liste en deux listes *l1*, *l2* avec $List.length\ l1 = List.length\ l2 + d$, d étant 0 ou 1.

Elle est définie par induction sur son argument. Une simple induction permet de voir qu'elle vérifie sa spécification. Le cas où l est vide est clair et dans le cas non-vide, si on a

$$List.length\ l1 = List.length\ l2 + d$$

alors

$$List.length\ (x :: l2) = List.length\ l1 + (1 - d)$$

et si d vaut 0 ou 1, 1-d vaut 0 ou 1.

```
let rec decoupe = function
  | [] -> [], []
  | x :: r -> let l1, l2 = decoupe r in x :: l2, l1
```

2. Une seconde fonction *fusion* fusionne deux listes supposées triées, c'est-à-dire construit une liste triée contenant les même éléments.

```
let rec fusion = function
  | [], l2 -> l2
  | l1, [] -> l1
  | (x1 :: r1 as l1), (x2 :: r2 as l2) ->
      if x1 <= x2 then x1 :: fusion (r1, l2) else x2 :: fusion (l1, r2)
```

3. La fonction réalisant le tri, *tri_fusion*, applique alors la méthode "diviser pour régner" : elle appelle *decoupe* pour séparer la liste donnée en deux listes de même taille, les trie puis fusionne les résultats avec *fusion* .

```
let rec tri_fusion = function
  | [] | [_] as l -> l
  | l -> let (l1, l2) = decoupe l in fusion (tri_fusion l1, tri_fusion l2)
```

Module Frequence

4. Fréquence des dix mots apparaissant le plus dans un texte.

```
open Printf
```

5. Limite inférieure sur le nombre de lettres dans un mot.

```
let limite = ref 2
```

```
let _ = if Array.length Sys.argv > 1 then limite := int_of_string Sys.argv.(1)
```

6. Table contenant le nombre d'occurrences de chaque mot.

```
let table = Hashtbl.create 97
```

7. Lecture des mots.

```
let rec lire () =
  let m = read_line () in
  if String.length m ≥ !limite then begin
    try incr (Hashtbl.find table m)
    with Not_found → Hashtbl.add table m (ref 1)
  end;
  lire ()
```

8. Tableau contenant les dix mots apparaissant le plus, triés par nombre d'occurrences (le mot apparaissant le plus à l'indice 0).

```
let frequences = Array.create 10 (0,"")
```

9. Affichage des résultats.

```
let afficher () =
  Array.iter (fun (f, m) → if f ≥ 1 then printf "%5d␣%s\n" f m) frequences
```

10. Calcul des résultats.

```
let resultats () =
  let insertion m r =
    let f = !r in
    (* on cherche à placer le mot à l'indice i qui est une place libre *)
    let rec insere i =
      if i = 0 ∨ fst frequences.(i − 1) > f then begin
        (* on a trouvé la bonne place *)
        if i < 10 then frequences.(i) ← (f, m)
      end else begin
        (* sinon on décale et on recommence *)
        if i < 10 then frequences.(i) ← frequences.(i − 1);
        insere (i − 1)
      end
    in
    insere 10
  in
  Hashtbl.iter insertion table;
  afficher ()
```

11. Programme principal.

```
let _ = try lire () with End_of_file → resultats ()
```