TD n°10 - Structure de données sac

1.
```
let exemple_sac() =
  let s = creer() in
  mettre s 0;
  mettre s 0;
  mettre s 1;
  mettre s 2;
  mettre s 4;
  mettre s 5;
  mettre s 10;
  mettre s 23;
  s;;
```

2. Pour compter combien d'éléments il contient, on est obligés de vider le sac. Il faut qu'on puisse stocker les éléments dans une structure intermédiaire afin de pouvoir les remettre dans le sac. Par exemple on peut les mettre dans un autre sac.

```
let taille s =
  let s2 = creer () in
  let res = ref 0 in
  while not (est_vide s) do
    mettre s2 (prendre s);
    res:=!res+1
  done;

  (*Rétablir le sac vdash état originel*)
  while not (est_vide s2) do
    mettre s (prendre s2);
  done;
  !res;;
```

3. On veut à la fois que la structure soit mutable, mais aussi qu'elle augmente de taille. On va donc choisir une ref de liste.

```
let creer () =
  ref [];;

let est_vide s = !s=[];;

let prendre s = match !s with
  |[] -> failwith "le sac est vide"
  |t::q -> s:= q; t;;

let mettre s x = s:=x::!s;;
```

4. La taille du sac est la taille de la liste.

```
let taille2 s =
  List.length !s;;
```

5. Comparer la complexité des deux fonctions `taille` et `taille2`.
prendre et mettre coutent un $O(1)$. Enlever et remettre tous les éléments coute un $O(n)$.
`List.length` coute $O(n)$.