Correction TP 36

Algorithme de Lempel-Ziv-Welsh
Décompression

Exercice 36.1 C'est le même principe que pour `output_code`, en retournant le procédé...

```
let input_code f d =
  let acc = ref 0 in
  for i = 0 to d - 1 do
    let bit = input_bit f in
    if bit then acc := !acc + (1 lsl i)
  done;
  !acc
```

Je donne tout de suite une autre petite fonction utilitaire qui permet d'écrire tous les octets qui compose une chaîne de caractère `s` dans un fichier `f` :

```
let output_string f s =
  for i = 0 to String.length s - 1 do
    output_byte f (int_of_char s.[i])
  done
```

1 Décompression pour les petits

Exercice 36.2 On vérifie directement qu'on trouve le message : `ABABABA`

Exercice 36.3

```
let lzw_decomp_naif table d entree sortie =
  let f_in = open_in_bits entree in
  let f_out = open_out_bin sortie in
  try
    while true do
      let code = input_code f_in d in
      let c = table.(code) in
      output_string f_out c
    done
  with
  | End_of_file ->
    close_in_bits f_in;
    close_out f_out
```

11 Décompression pour les grands

Exercice 36.4

```
let lzw_decomp d entree sortie =
  let f_in = open_in_bits entree in
  let f_out = open_out_bin sortie in
  let t = Array.make (1 lsl d) "" in
  for i = 0 to 255 do
    let c = string_of_byte i in
    t.(i) <- c
  done;
  let table_size = ref 256 in
  let c = ref (input_code f_in d) in
  output_byte f_out !c;
  try
    while true do
      let n = input_code f_in d in
      let x = if n = !table_size then t.(!c).[0] else t.(n).[0] in
      if !table_size < 1 lsl d
      then (
        t.(!table_size) <- t.(!c) ^ String.make 1 x;
        incr table_size);
      output_string f_out t.(n);
      c := n
    done
  with
  | End_of_file ->
    close_in_bits f_in;
    close_out f_out
```