Test : Pour tester nos fonctions, on a essayé d'abord avec un fichier de petite taille contenant la chaîne suivante : «aaafpakefk », notre programme construit bien l'arbre huffmann codée et l'affiche comme suivant :

```
f203-24:~/Bureau/Apnées/Algo/huffman
[09:53:05]taguia$ ./huff_encode test.txt out.txt
Occuerences du caractere
 (code 10) : 1
Occuerences du caractere a (code 97) : 4
Occuerences du caractere e (code 101) : 1
Occuerences du caractere f (code 102) : 2
Occuerences du caractere k (code 107) : 2
Occuerences du caractere p (code 112) : 1
#
  #
    k
    #


      e
  #
    #
      p
      f
    a
f203-24:~/Bureau/Apnées/Algo/huffman
[09:57:55]taguia$
```

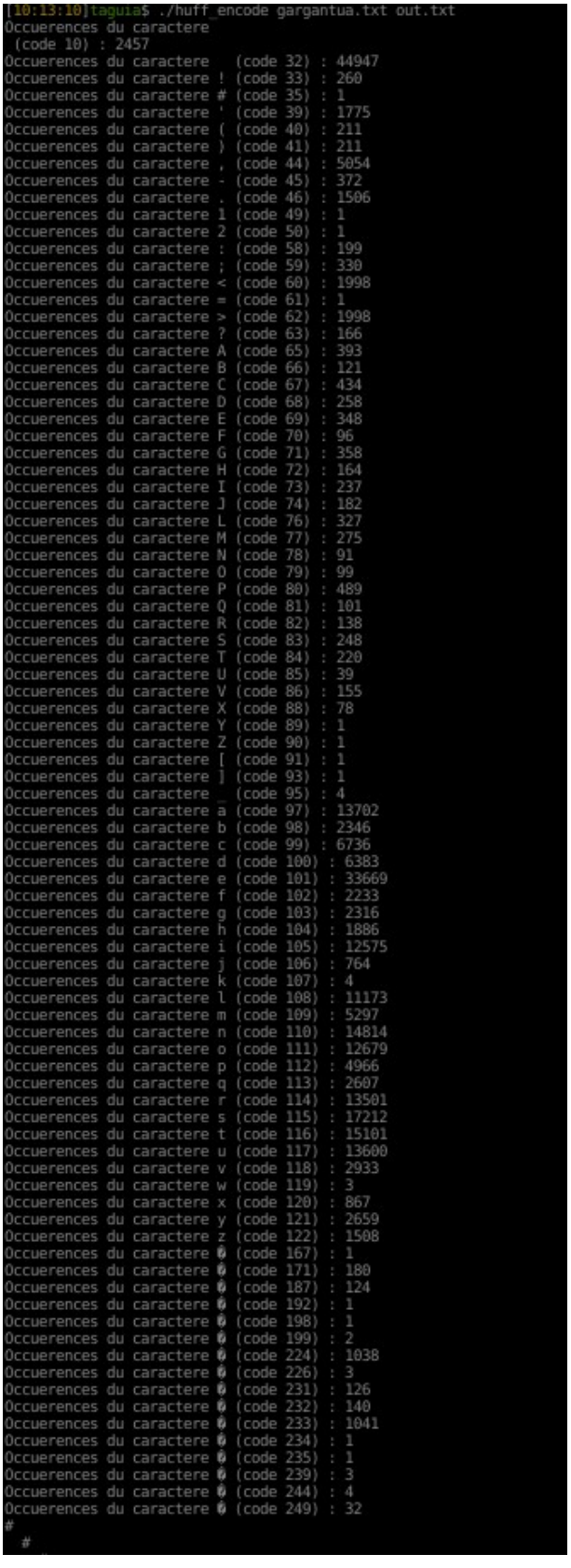
```
    a
f203-24:~/Bureau/Apnées/Algo/huffman
[09:57:55]taguia$ ./huff_decode out.txt
entier lu : 107
fg : 0
fd : 0
entier lu : 10
fg : 0
fd : 0
entier lu : 101
fg : 0
fd : 0
entier lu : 256
fg : 10
fd : 101
entier lu : 257
fg : 107
fd : 256
entier lu : 112
fg : 0
fd : 0
entier lu : 102
fg : 0
fd : 0
entier lu : 258
fg : 112
fd : 102
entier lu : 97
fg : 0
fd : 0
entier lu : 259
fg : 258
fd : 97
entier lu : 260
fg : 257
fd : 259
a
  a
    k
    a


      e
  a
    a
      p
      f
    a
```

Et si on le décode, on voit bien qu'on obtient le même arbre .

Remarque : on choisit comme racine le symbole # pour l'encodage et lors du décodage, le symbole # se change en a car dans les fichiers fournis, tout ce qui n'est pas un arbre en a.

On a fait des test avec des fichiers plus volumineux (candide.txt, gargantua.txt, huff_encode.c, huff_decode.c, tout les fichiers fournis).



L’utilisation des modules :

Pour encoder nos textes, on utilise les structures d’arbres, file-à-priorité et pour décoder on utilise les arbres et les bfiles car l’algorithme d’huffmann encore chaque lettre avec des 0 et des 1, donc pour le décoder il fallait le lire bit-à-bit.

Conclusion :

Au début, on encodait nos fichiers de la sorte «fprintf(fichier_encode,"%i",HuffmanCode[(int)c].code[i])»,
et ça faisait le travail demander avec des fichiers de petites tailles et dés qu’on testait avec des fichiers plus volumineux, on obtenait une segmentation fault. Aprés le débogage, on s’est apperçu qu’il fallait caster le caractère lu par le type Element et non pas par int «fprintf(fichier_encode,"%i",HuffmanCode[(Element)c].code[i])».