Compression

1. Introduction :

Exercice 1

Une vidéo codée au format 480*p* est composée d'images dont la dimension est 720 ×480 pixels. Sachant que :

- un film d'une heure a été enregistrée avec une fréquence d'image de 30 images par seconde *(30 fps)*
- que la couleur d'un pixel est codée sur 3 octets *(1 octet pour le rouge, 1 octet pour le vert, 1 octet pour le bleu)*

1. Quel est la taille en octets de ce film?
2. Cela vous semble-t-il réalisable?

2. Décompression :

Exercice 2

On considère le texte ci-dessous :

```
ATX, aTX
Au soleil, sYs la pluie,
Zdi ou Znuit
Il y a tYt ce que vYs vYlez aTX
```

Avec la table de correspondance :

W ⇝ "ux "	X ⇝ "Champs-Elysées"
Y ⇝ "ou"	Z ⇝ "à mi"

1. Comptez le nombre de caractères *(en y incluant les espaces et les retours à la ligne)* du texte et le nombre de caractères utilisés dans la table de correspondante.
2. a. Reconstruire, dans le cadre ci-dessous, le texte en remplaçant les caractères W, X, Y, Z par leurs correspondances ;

```
A........., a......
Au soleil, s.....s la pluie,
.....di ou .....nuit
Il y a t...t ce que v...s v...lez a...
```

   b. Combien de caractères apparaissent maintenant dans le texte reconstitué?
   Déterminer le taux de compression *(exprimé en pourcentage)*

3. a. Quelle modification peut-on faire pour améliorer pour améliorer la compression de ce texte?
   b. Pour quelle raison, l'introduction de la correspondance V ⇝ *ui* n'est pas pertinente?

3. Première compression :

Exercice 3

extrait de csunplugged.org
La séquence "`sifflant soufflant`" peut se coder en `siX souX` lorsqu'on garde en mémoire que `X` correspond à `fflant`.

Ainsi :

- `sifflant soufflant` est codée sur 18 octets
- `siX souX` est codée sur 15 octets *(8 pour la chaine de caractères et 7 pour la table de correspondance)*.

On considère le texte suivant :

```
Vive le vent, vive le vent
Vive le vent d'hiver
Qui s'en va sifflant soufflant
dans les grands sapins verts
```

1. Combien d'octets sont nécessaires pour enregistrer ce texte *(un caractère = 1 octet)*
2. a. Barrez les séquences d'au moins 2 caractères se répétant plusieurs fois dans le texte.
   b. Ecrire une table de correspondance permettant de mémoriser les séquences barrées.
   c. Quelle est alors la taille du texte compressé?

4. Différentes compressions :

Exercice 4

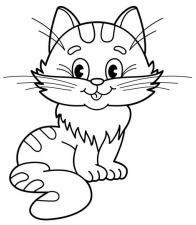
- Le format `JPEG` *((acronyme de Joint Photographic Experts Group)* est une norme qui définit le format d'enregistrement et l'algorithme de décodage pour une représentation numérique compressée d'une image fixe. C'est un processus de compression avec perte.
- Le format `PNG` *(Portable Network Graphics)* est un format ouvert d'images numériques, qui a été créé pour remplacer le format GIF, à l'époque propriétaire et dont la compression était soumise à un brevet. Le PNG est un format sans perte spécialement adapté pour publier des images simples comprenant des aplats de couleurs.

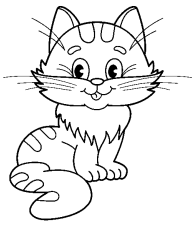
`Wikipedia.fr`

Voici l'exemple de deux compressions :

jpg - 15,6ko

png - 316ko


jpg - 34,9ko


png - 6,4ko

1. Lire l'encadré ci-dessus
2. Pouvez vous dire dans quel cas, il est préférable d'utiliser le format JPG ou le format PNG?

(cc) BY-NC

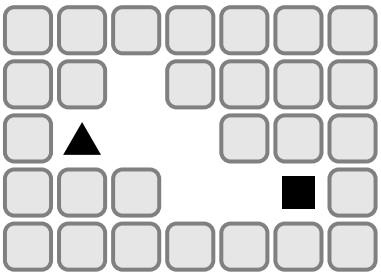
3. Parmi ces deux formats, pour un processus de compression, puis de décompression, lequel permet de retrouver exactement l'image de départ?

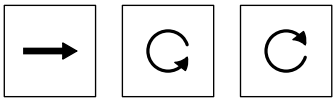
5. Turtle :

Exercice 5

La flèche doit suivre le chemin pour rejoindre le carré :

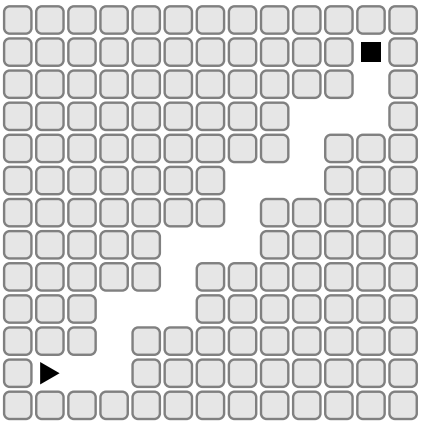
Utiliser les actions suivantes pour effectuer le parcours :





Exercice 6

La flèche doit suivre le chemin pour rejoindre le carré :

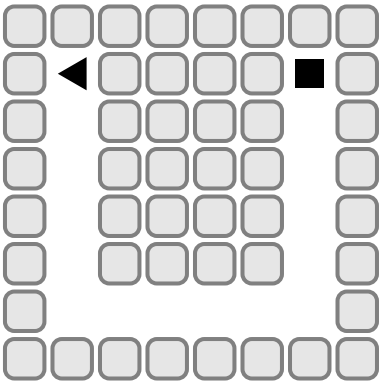


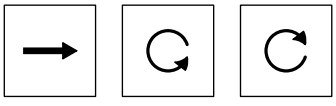
Utiliser une boucle et le minimum d'actions pour effectuer le parcours :

Exercice 7

La flèche doit suivre le chemin pour rejoindre le carré.

Utiliser deux boucles et le minimum d'actions pour effectuer le parcours :





6. Algorithme d'Huffman :

http ://www-igm.univ-mlv.fr/ lecroq/cours/huff.pdf

La chaîne "*cagataagagaa*" se code dans la table ASCII en héxadécimal :

c	a	g	a	t	a	a	g	a	a	a	FIN
63	61	67	61	74	61	61	67	61	61	61	00

(On remarquera le dernier caractère pour coder la fin de la chaîne de caractère)

Sa transcription en bits s'écrit donc sur 96 bits :

```
110001111000011100111110000111101001100001
110000111001111100001110011111000011100001
```

L'algorithme d'Huffman se base sur la fréquence d'apparition de chacune des lettres de cette chaîne de caractères :

Caractères	c	a	g	t	FIN
Nombre d'apparition	1	7	2	1	1

De manière astucieuse, une séquence de bits est associée à chacun des caractères :

a⇝1 ; c⇝0000 ; g⇝01 ; t⇝0001 ; FIN⇝001

Cette association est faite, par lecture de la séquence de bits de gauche à droite et dès qu'une séquence est trouvé, elle est remplacé par le caractère correspondant. La chaîne codée devient :

000010110001110110111001 *(codage sur 24 bits)*

qui se lit :

0000	1	01	1	0001	1	1	01	1	01	1	1	001
c	a	g	a	t	a	a	g	a	g	a	a	FIN

1. Une chaîne de caractères a été compressée par l'algorithme d'Huffman en la séquence :
   101110111110001000111000

   La table de correspondance associée à ce codage est :
p⇝100 ; i⇝11 ; k⇝011 ; é⇝010 ; d⇝001 ; w⇝101 ; a⇝000

   En lisant de gauche à droite la plus grande séquence de bits à utiliser, retrouver le mot caché.

2. On souhaite compresser la chaîne de caractère *rareradar*.

   a. La table de correspondance donnée par l'algorithme d'Huffman est :
   r⇝0 ; a⇝11 ; e⇝100 ; d⇝101
   Sur combien de bits ce mot a été compressé?

   b. Le choix de la table de correspondance est essentielle : Que se passe-t-il si vous compressez puis décompressez la chaîne de caractères "*rareradar*" avec la table de correspondance :
   r⇝0 ; a⇝1 ; e⇝10 ; d⇝11

(cc) BY-NC