WebPédago
Partagez la connaissance !

BAC GÉNÉRAL 2025
Correction épreuve de spécialité numérique et sciences informatiques
18 juin 2025

Exercice 1

Partie A

1.
Le mot binaire à utiliser est « 010 ».

2.
Le texte codé est « espion »

3.
Pour obtenir les symboles classés par taille d'encodage croissante, il faut effectuer un parcours en largeur : les feuilles les plus hautes, correspondant aux encodages les plus courts, seront ainsi parcourues en premier.

Partie B

4.
Le total d'occurrences dans la phrase à encoder est 22, soit 2 fois 11. On peut alors scinder le tableau présenté en deux parties, celles présentées à la figure 2. On obtient 2 tableaux triés dont chacun présente une somme de 11. Le tableau de départ est la racine de l'arbre, son fils gauche, étiqueté par un 1, la première partie, et son fils droit, étiqueté par un 0, la deuxième.

5.
L'arbre que l'on obtient est de hauteur 5. C'est la plus longue séquence de bits nécessaire pour encoder un symbole du message.

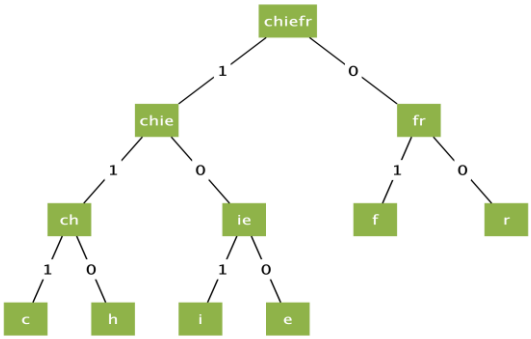
6.
En encodant la phrase en ASCII, on a besoin de 22 octets, soit 176 bits. Avec la méthode de Shannon-Fano, le tableau suivant, tiré de l'arbre, nous permet de compter le nombre de bits nécessaires pour encoder la phrase :

symbole	e	_	s	j	n	p	,	d	o	c	u	i
nombre de bits	2	3	3	4	4	4	4	5	5	4	4	4

En multipliant le nombre d'occurrence de chaque lettre par le nombre de bits nécessaires pour l'encoder, on arrive à 75 bits, soit moins de 10 octets, ce qui est environ deux fois moins qu'en ASCII.

7.

L'arbre attendu est le suivant :



Partie C

8.

```
08 dico[symbole] = dico[symbole] + 1
10 dico[symbole] = 1
```

9.

```
def somme_occ(tab):
    somme = 0
    for t in tab:
        somme += t[1]
    return somme
```

10.

```
09 return "1" + shannon(symbole, t1)
11 return "0" + shannon(symbole, t2)
```

11.

La fonction se termine car elle s'arrête lorsque le tableau n'a plus qu'un élément et que chaque appel récursif passe en paramètre un tableau de taille strictement plus petite que tab, (car séparé par la fonction separe en deux tableaux non-vides).

12

```
def encode_shannon(texte):
    dico = creer_dico_occ(texte)
    tab_trie = creer_tab_trie(dico)
    resultat = ""
    for lettre in texte:
        resultat += shannon(lettre, tab_trie)
    return resultat
```

Exercice 2

1.
Les valeurs prises par un attribut qui est une clé primaire doivent être uniques. Or deux adhérents peuvent avoir le même nom de famille, donc l'attribut nom ne peut convenir en tant que clé primaire.

2.
Cette requête renvoie la table extraite de la table jeu dont les colonnes sont nomJeu et editeur, en classant les jeux par leur nom dans l'ordre alphabétique.

3.
```
SELECT nomJeu
FROM emprunt
WHERE dateRendu IS NULL ;
```

4.
```
SELECT nom, prenom
FROM adherent
JOIN emprunt ON emprunt.idAdherent = adherent.idAdherent
WHERE nomJeu = « Catan » ;
```

5.
```
UPDATE emprunt
SET dateRendu = « 2025-06-03 »
WHERE idEmprunt = 1538 ;
```

6.
```
SELECT nomJeu, categorie
FROM jeu
WHERE anneeSortie ≥ 2010 AND ageMinimum < 10 ;
```

7.
adherent sera une clé étrangère faisant référence à la clé primaire id_adherent de la relation adherent.
Evenement sera une clé étrangère faisant référence à la clé primaire nom de la relation evenement.

8.
On suppose que la variable liste a la même valeur à la fin du script précédent.
```
dict_emprunts = dict()
for jeu in liste:
    if jeu in dict_emprunts:
        dict_emprunts[jeu] = dict_emprunts[jeu] + 1
    else:
        dict_emprunts[jeu] = 1
```

9.

```
def le_podium(dico): podium = [[], [], []]
    scores_podium = [0, 0, 0]

    for i in range(3):
        for jeu, score in dico.items():
            if score > scores_podium[2-i]  and score not in scores_podium:
                print(score)
                scores_podium[2-i] = score

    for i in range(3):
        for jeu, score in dico.items():
            if score==scores_podium[i]:
                podium[i].append(jeu)

    return podium
```

Exercice 3

Partie A

1.
L encodé par E donne le nombre 11+4 = 15, correspondant à P.
De même pour les autres lettres, le message encodé est PGRDX.

2.
```
def indice(L, element):
    i = 0
    for i in range(len(L)):
        if L[i] == element:
            return i
```

3.
```
def lettres_vers_indices(chaine):
    resultat = []
    for lettre in chaine:
        resultat.append(indice(alphabet, lettre))
    return resultat
```

4.
```
for k in range(n):
    ind = indices_msg[k] + indices_cle[k]
    if ind >= 26:
        ind = ind - 26
    indices_msg_chiffre.append(ind)
msg_chiffre = indices_vers_lettres(indices_msg_chiffre)
return msg_chiffre
```

5.
Comme la longueur de la clé est inférieure à la longueur du message à chiffrer, une exception AssertionError est levée lors de l'exécution de la ligne 2.

6.
On obtient BRAVO en déchiffrant.

7.
```
indices_msg_dechiffre = []
for k in range(n):
    ind = indices_msg[k] - indices_cle[k]
    if ind < 0:
        ind = ind + 26
    indices_msg_dechiffre.append(ind)
msg_dechiffre = indices_vers_lettres(indices_msg_dechiffre)
return msg_dechiffre
```

8.
```
indices_msg_chiffre = []
```

```
for k in range(n):
    ind = lettres_vers_indices(msg[k]) - lettres_vers_indices(cle[k])
    if ind < 0:
        ind = ind + 26
    indices_msg_chiffre.append(ind)
msg_chiffre = indices_vers_lettres(indices_msg_chiffre)
return msg_chiffre
```

Partie B

9.
Avec un algorithme de chiffrement symétrique, une unique clé sert à chiffrer et à déchiffrer. Avec un algorithme de chiffrement asymétrique, il y a une clé publique pour chiffrer et une clé privée pour déchiffrer.

10.
Bob doit utiliser sa clé privée pour déchiffrer le message d'Alice.

11.
On appelle ceci « attaque de l'homme du milieu ». Nous l'appellerons Chris.
Imaginons que Chris intercepte les conversations d'Alice et Bob.

- Lorsque Bob donne sa clé publique à Alice, Chris l'intercepte et donne sa propre clé publique à Alice.
- Lorsqu'Alice renvoie son message chiffré à Bob, Chris l'intercepte et déchiffre l'information d'Alice avec sa propre clé privée.
- Finalement, Chris chiffre le message « en clair » d'Alice avec la clé publique de Bob, et envoie cela à Bob, qui ne se doute de rien.

12.
Si A veut communiquer avec B, alors B envoie sa clé publique signée numériquement, ce qui tient lieu de certificat. A vérifie auprès d'un site de certification que la signature donnée est valide et correspond bien à B. Le but de cette étape est d'éviter une situation décrite dans la question précédente.
Ensuite A qui se sert de la clé publique de B pour chiffrer une clé symétrique et lui envoyer. B la déchiffre et le reste de la communication est effectué en chiffrement symétrique.

13.
On procède de cette manière

- pour éviter une attaque de l'homme du milieu ;
- car le chiffrement asymétrique est bien plus lent que le chiffrement symétrique et que l'on veut privilégier la vitesse de communication.

Partie C

14.
Marc s'est trompé et a entré une mauvaise IP. Il faut qu'il entre ping 192.168.110.115 .

15.
Ce masque de sous réseau s'écrit 255.255.255.224.

16.

Puisqu'il reste 5 bits libres pour les parties hôte des adresses IP de ce réseau, en enlevant l'adresse du réseau et l'adresse de diffusion, $2^5 - 2 = 30$ adresses IP peuvent être attribuées.

17.
134 = 128 + 4 + 2 donc sa représentation binaire est 1000 0110.

18.
L'affichage obtenu indique que les paquets ont été correctement transmis : le poste de travail de
Zoé fait donc partie du même sous-réseau que le poste qu'elle essaie de joindre. Avec le choix de
l'administratrice système, il suffit d'appliquer le masque sur le dernier octet des adresses IP pour réussir à identifier les deux postes qui font partie du même sous-réseau (les trois premiers octets seront toujours identiques).
En appliquant le masque de sous-réseau sur le dernier octet de l'adresse IP de Zoé, on obtient 10000000.

```
   1 1 1 0 0 0 0 0 (224)
ET 1 0 0 0 0 1 1 0 (134)
-------------------------
   1 0 0 0 0 0 0 0 (128)
```

On obtient le même résultat pour le dernier octet de l'adresse IP du poste de travail de Marc :

```
   1 1 1 0 0 0 0 0 (224)
ET 1 0 0 1 1 0 0 1 (153)
-----------------------
1 0 0 0 0 0 0 0 (128)
```

Pour le poste de travail de Bob, le résultat est donné dans l'énoncé :
01100000 (96) .
Les postes de travail de Zoé et Marc font donc partie du même sous-réseau. On en déduit que Zoé a exécuté la commande n°2.