ÉCOLE POLYTECHNIQUE
ÉCOLE SUPÉRIEURE DE PHYSIQUE ET CHIMIE INDUSTRIELLES

CONCOURS D'ADMISSION 2008

FILIÈRE MP - OPTION PHYSIQUE ET SCIENCES DE L'INGÉNIEUR FILIÈRE PC

COMPOSITION D'INFORMATIQUE
(Durée : 2 heures)

L'utilisation des calculatrices n'est pas autorisée pour cette épreuve.

Ave Cesar (zud bdrzq)

On cherche à crypter un texte t de longueur n composé de caractères en minuscules (soit 26 lettres différentes) représentés par des entiers compris entre 0 et 25 (0 ↔ a, 1 ↔ b, . . . 25 ↔ z). Nous ne tenons pas compte des éventuels espaces.

Ainsi, le texte ***ecolepolytechnique*** est représenté par le tableau suivant où la première ligne représente le texte, la seconde les entiers correspondants, et la troisième les indices dans le tableau t.

e	c	o	l	e	p	o	l	y	t	e	c	h	n	i	q	u	e
4	2	14	11	4	15	14	11	24	19	4	2	7	13	8	16	20	4
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

Ce codage est le plus rudimentaire que l'on puisse imaginer. Il a été utilisé par Jules César (et même auparavant) pour certaines de ses correspondances. Le principe est de décaler les lettres de l'alphabet vers la gauche de 1 ou plusieurs positions. Par exemple, en décalant les lettres de 1 position, le caractère a se transforme en z, le b en a, ... le z en y. Le texte ***avecesar*** devient donc ***zudbdrzq***.

Le but de ce problème est donc d'implémenter la méthode de césar qui code et décode une chaine de caractère avec une clé k quelconque (désignant le décalage des lettres de k positions).

Question 1. Ecrire la fonction ***getAlphabet()*** qui renvoie l'alphabet des lettres minuscules (soit 26 lettres différents), sous forme d'une suite de chaine de caractères séparés par des virgules.

Question 2. Ecrire la fonction ***cesar_car(lettre,k, alphabet)*** qui crypte une lettre de l'alphabet par la clé k. on laisse inchangés les lettres hors de l'alphabet (espace, ponctuation, minuscule, etc).

Exemple : Pour lettre = 'e', k= 3 on renvoie 'h' et pour lettre = 'z' et k = 3, on renvoie 'c', etc.

Question 3. Ecrire la fonction ***cesar_crypte_mot(mot,k, alphabet)*** qui crypte un mot par la clé k.

Question 4. En déduire la fonction ***cesar_decrypte_mot(mot,k, alphabet)*** qui décrypte un mot selon la clé k.

Pour réaliser le décodage, il faut connaître la valeur du décalage. Une manière de la déterminer automatiquement est d'essayer de deviner cette valeur. L'approche la plus couramment employée est de regarder la fréquence d'apparition de chaque lettre dans le texte crypté. En effet, la lettre la plus fréquente dans un texte suffisamment long en français est la lettre '**e'**.

Question 5. Ecrire la fonction ***frequence(mot,alphabet)*** qui retourne un dictionnaire de fréquence de chaque lettre de l'alphabet, présente dans le mot (les clés représentent les lettres et les valeurs du dictionnaire représentent leurs fréquences) .

Question 6. Écrire la fonction ***decodageAuto(phraseCr)*** qui prend en argument une phrase française représentant le texte crypté ; et qui retourne la phrase d'origine.

Codage de Vigenère

Au XVIème siècle, Blaise de Vigenère a modernisé le codage de César très peu résistant de la manière suivante. Au lieu de décaler toutes les lettres du texte de la même manière, on utilise un texte clé qui donne une suite de décalages.

Pour crypter un texte, on décale la première lettre du texte d'un rang correspondant à celui de la première lettre de la clé dans l'alphabet et ainsi de suite. Par exemple, si la clé est « python », la première lettre sera décalée de 15 (p est la 15ème lettre de l'alphabet, en partant de 0), le deuxième de 24 etc. On revient au début de la clé si le texte à crypter est plus long que la clé).

Exemple :

La clé est : python

Texte clair : 'concours prepa mp pc pt'

Texte crypté : 'rmgjchgq wfrey td ea wh'

Question 7. Ecrire la fonction ***viginere_crypt_mot(mot , cle)*** qui crypte un mot par la clé « cle » selon la méthode de Vigenére.

Question 8. Ecrire la fonction Ecrire la fonction ***viginere_decrypt_mot(mot , cle)*** qui décrypte un mot par la clé « cle » selon la méthode de Vigenére.

Question 9. Ecrire la fonction ***viginere_file(f_in , f_out , cle)*** qui crypte le contenu du fichier f_in et l'écrire dans un fichier f_out.

Codage RSA

RSA est une méthode de cryptographie asymétrique .Elle pallie ce problème en distinguant la clé avec laquelle le message est chiffré (la clé publique) et celle avec laquelle on déchiffre le message codé (la clé privée). On peut voir la clé publique comme l'adresse d'une boite postale connue un peu partout, et la clé privée comme la clé de cette boite.

Création des clés

Avant de pouvoir recevoir des messages codés avec RSA, il faut posséder une clé publique, et la clé privée correspondante. On procède comme suit :

On choisit deux nombres premiers au hasard p et q (plus ils sont grands, plus le cryptage est sûr, mais plus le temps de calcul est long)

Question 10. Ecrire la fonction est_premier(n) qui vérifie si n est premier

Question 11. Ecrire la fonction nombres_premiers(a,b) qui renvoie une liste de nombres premiers entre a et b inclus.

Question 12. Ecrire la fonction choisir_premier(a,b) qui renvoie au hasard deux entiers premiers p et q entre a et b inclus.

Note : On peut utiliser la fonction randint(a,b) du module random qui renvoie un entier au hasard dans l'intervalle [a,b].

Et, pour créer les deux clés publique et privé, on procède comme suit :

- On calcule n =pq et $\phi = (p-1)(q-1)$
- On choisit un entier e (l'exposant d'encryptage) au hasard tel que :
  - $p,q < e < \phi$
  - pgcd (e, ϕ) = 1

  Note : **<u>on peut choisir un entier premier et qui ne divise pas ϕ</u>**

Question 13. Ecrire une fonction cle_publique(p,q) qui renvoie le couple (e,n) , la clé publique de cryptage.

Enfin, on choisit d (la clé privé), un entier tel que **e x d = 1 mod ϕ** , à l'aide des coefficients de Bezout : u, v et le pgcd(a,b), tels que au + bv = pgcd(a, b) , selon le principe suivant :

On définit deux suites (x_i), (y_i) qui vont aboutir aux coefficients de Bezout.

L'initialisation est : $x_0 = 1$, $x_1 = 0$, $y_0 = 0$, $y_1 = 1$

et la formule de récurrence pour i > = 1 :

$$x_{i+1} = x_{i-1} - q_i x_i$$

$$y_{i+1} = y_{i-1} - q_i y_i$$

où q_i est le quotient de la division euclidienne de a_i par b_i

, les coefficient u,v de Bezout correspond au terme x_i , y_i

Question 14. Ecrire la fonction euclide(a,b) qui renvoie le pgcd(a,b) selon le principe d'Euclide : pgcd(a, b) = pgcd(b, a (mod b)) et pgcd(a,0) = a.

Question 15. Ecrire la fonction euclide_etendu(a,b) qui renvoie les coefficients u,v de Bezout

Question 16. En déduire la fonction cle_privee(p,q,e) qui renvoie la clé privé d'encryptage (d,n), avec d est l'inverse de e modulo phi.

Note : Si e * d + phi * v = pgcd(e,phi) = 1 alors d est un inverse de e modulo phi.

Une fois ces étapes accomplies, on dispose d'une clé publique : le couple (e,n), et d'une clé privée, le couple(d, n).
Voyons maintenant comment on se sert de ces clés pour crypter des entiers

Cryptage et décryptage.
On suppose disposer d'un message sous la forme d'un entier M
On définit alors le cryptage du message M par la clé publique(e,n) comme $C = M^e \bmod n$.
On retrouve l'original M à partir de C et de la clé privée (d,n) par $M = C^d \bmod n$.

Question 17. Ecrire la fonction codage_rsa(m,n,e) qui renvoie le cryptage de l'entier m par la clé publique (n, e).

Question 18. Ecrire la fonction decodage_rsa(x,n,d) qui renvoie le décryptage de l'entier x par a clé privée (d et n).