TP 4

Conversion entre différentes bases.

Exercice 1

1) Écrire une procédure `conversion_base(n,b)`, prenant en entrée deux nombres entiers `n` et `b`, et qui renvoie la liste des chiffres de `n` en base `b`. En d'autres termes cette procédure renvoie un tableau `t` de `m` entiers compris entre `0` et `b-1` tel que

$$n = \sum_{k=0}^{m-1} t[m-k+1]b^k.$$

Par exemple, `chiffres_base(421,10)` doit renvoyer `[4,2,1]` ; `conversion_base(42,2)` doit renvoyer `[1,0,1,0,1,0]` ; `chiffres_base(510,16)` doit renvoyer `[1,15,14]`.

2) Écrire une procédure `entier_base(t,b)` qui calcule l'entier tel que ses chiffres en base `b` soient listés par `t`. Il s'agit de la procédure inverse de la précédente.

Exercice 2

Écrire une procédure `conversion_bases(t,b1,b2)` qui renvoie le tableau qui liste les chiffres en base `b2` du nombre dont les chiffres en base `b1` sont listés par `t`.

Exercice 3: Somme des chiffres

1) Écrire une procédure `somme_chiffres_base(n,b)` qui renvoie la somme des chiffres de `n` en base `b`. Cette procédure n'utilisera pas de listes.

2) Écrire une procédure `chiffre_associe_base(n,b)` qui répète la fonction précédente jusqu'à obtenir un chiffre compris entre 1 et `b-1`.

Par exemple, calculons `chiffre_associe_base(8643,10)`. La somme des chiffres en base 10 de `8643` vaut `8+6+4+3=21`. Le nombre `21` n'est pas réduit un seul chiffre, on continue donc l'opération : `2+1=3`. Le nombre `3` étant un chiffre compris entre 1 et `b-1`, c'est le résultat souhaité. L'exéution de `chiffre_associe_base(8643,10)` doit donc renvoyer `3`.

3) Écrire une fonction qui prend en paramètres trois entiers `m`, `n` et `b`, et qui calcule l'entier :

```
chiffre_associe_base(
        chiffre_associe_base(m,b) *
        chiffre_associe_base(n,b),
        b
) - chiffre_associe_base(m*n,b)
```

Que renvoie en réalité cette fonction ?

Encodage des lettres et cryptographie de base.

Exercice 4: Encodage des lettres

1) Écrire une procédure `position_lettre(l)` qui calcule la position de la lettre `l` dans l'alphabet. Par exemple `position_lettre('E')` renverra `5`. Les plus malins utiliseront la fonction `ord` qui transforme un caractère en son code ASCII correspondant. (La tester pour voir...)
2) Écrire une procédure `nombre_en_lettre(n)` qui renvoie la lettre qui est en position `n` dans l'alphabet. Par exemple `nombre_en_lettre(25)` renverra `'Y'`. Les plus malins utiliseront la fonction `chr` qui renvoie un caractère selon son code ASCII. (La tester pour voir...)
3) Écrire une procédure `texte_en_nombres(st)` qui renvoie le tableau listant dans l'ordre les positions des lettres qu'on peut trouver dans la chaîne de caractères `st`. Par exemple, `texte_en_nombres("BYE BYE!")` renverra `[2,25,5,2,25,5]`.

   Aide : `list(st)` renvoie un tableau listant tous les caractères de `st`. Tester par exemple `list("Salut toi.")`.
4) Écrire une procédure `nombres_en_texte(t)` qui renvoie la chaîne de caractères composée des lettres dont les positions sont listées par `t`.

   Par exemple, `tableau_en_texte([2,25,5,2,25,5])` renverra `"BYEBYE"`.

Exercice 5: Décalage de lettres (Chiffre de César)

1) Écrire une procédure `decalage(st,d)` qui renvoie la chaîne de caractère obtenue à partir de `st` (qui également une chaîne de caractères) par un décalage de `d` positions de toutes ses lettres. On supposera que la lettre qui suit 'Z' est 'A'. On supprimera les caractères n'étant pas des lettres. Par exemple, `decalage("Oui !",10)` donnera `'YES'`.
2) Essayer de déchiffrer le message suivant obtenu à partir de la fonction précédente :

   'QJKTOSTRTLWXHZNFJTATEPIGDCYJVTUPBTJM'

Exercice 6: Chiffrement par substitution

1) Écrire une procédure `creation_alphabet(clef)` qui à partir de la chaîne de caractères `clef` uniquement composée de lettres toutes distinctes, renvoie le même mot où on a ajouté à la fin toutes les lettres qui n'étaient pas présentes dans ce mot, de manière à former une chaîne de 26 lettres. Par exemple, `creation_alphabet('CPBX')` renverra `'CPBXADEFGHIJKLMNOQRSTUVWYZ'`
2) Écrire une procédure `chiffrement(texte,clef)` qui chiffre la chaîne de caractère `texte` de la manière suivante :

   On calcule `creation_alphabet(clef)` qui est un mot de 26 lettres toutes distinctes : $a_1a_2\dots a_{26}$. On change toutes les occurrences de A dans `texte` par a_1, toutes les occurrences de B par a_2 etc... Par exemple, `chiffrement('CARTE','CPBX')` donnera 'BCQSA'.
3) Essayer de déchiffrer le message suivant, résultat de la fonction précédente où `clef` valait `'PYTHON'` :

   'DOQSCQNCOMHOUJSQ'

Exercice 7: Rajout aléatoire d'espaces

Afin que vos messages chiffrés soient plus visibles (et plus trompeurs ?), écrire une fonction qui rajoute aléatoirement des espaces à une chaîne de caractères donnée qu'on suppose uniquement constituée de lettres. On veillera à ne pas ajouter deux espaces consécutivement. Un flou est volontairement laissé sur la notion d'aléatoire, à vous de prendre des initiatives.

Exercice 8: Analyse de la fréquence

Écrire une fonction qui trie par ordre décroissant d'apparition les lettres apparaissant dans un texte donné. Pour information, en français, l'ordre d'apparition moyenne des lettres est le suivant : `EASINTRLUODCPMVGFBQHXJYZKW`

Exercice 9: Exercice défi

Essayer de déchiffrer le message suivant :

'BEBFS GERBJ YDEJS BJLJH QFDMS GBSQP FLREQ RRFSO FSLQB JCEHB BESLE
DFSQB JBBFD QCFDR LELRF SRBTE SLESD JYDEJ SQEMQ JBRLJ HRMJD QBEIF
SLJDR MSDEF DMEGS LESDB FSGQS LUHED ROESD KSHIT ELITJ HRJUE DRSLE
ERKSE BJPJH CEDIE QBHES WJRRH LJHRK SHREL EDMQH TJLMH MERLF SABEL
CFDAL ESUJY EMHRI ERJDH CJBGB EHDME LJYER SQELJ QITRH MERJR CLHRQ
HLELG FDMBJ YDEJS KSEUF RLECJ OEQRD EQECE RREGJ QEDIF BLECJ HQGBS
RRKSE BBEIF DQHML EKSEO ECEUJ QMQJB RLJDR MJDQB EIFSL JDRGB SQMEU
HDYRG JQJSM EQQFS QMEBB EERKS EGJLI FDQKS EDRED JSISD EPJFD OEDEG
SHQRL FSABE LQJAF HQQFD RSBJR LFSAB EQLEG LHRIE RREAR EILSE BBEER
OEQJH QKSEM ECFHR SCMHQ BJDGJ QQIFC CEDRB JSLJH QOEPJ HRQHO EDRJH
QGJQD LEGLH RBJYD EJSOE RERRE EDIFL CJCLE QHIED EQRRF HIEQR MFDIR
FDPLL EOEDE DJHGF HDRIE QRMFD IKSEB KSSDM EQRHE DQIJL UFSQD ECGJL
YDEZY SLEUF SQUFQ AELYE LQERU FQITH EDQFD CEBJM HRHBP JSRKS EOECE
UEDYE BMEQQ SQJSP FDMME QPFLR QBEBF SGBEC GFLRE ERGSH QBECJ DYEQJ
DQJSR LEPFL CEMEG LFIQ'