TD Maple 7

Le code bibinaire et un peu d'algèbre...

I - Coder en bibinaire

a. Qu'est-ce que c'est ?

Boby LAPOINTE, célèbre chanteur français, était aussi mathématicien à ses heures. Ayant trouvé le code binaire trop compliqué à utiliser, il inventa le code... bibinaire. Il suffit de remplacer les chiffres par des lettres. On commence par couper le nombre écrit en binaire en paquets de 2. S'il y a un nombre impair de chiffres, on rajoute un zéro à gauche, ce qui ne modifie pas la valeur de nombre. On commence par le premier groupe de deux chiffres le plus à droite. On remplace 00 par O, 01 par A, 10 par E, 11 par I.
Puis on prend le paquet de deux chiffres suivants en se déplaçant de droite à gauche. On remplace 00 par H, 01 par B, 10 par K, 11 par D.
Pour le paquet suivant, on recommence avec les voyelles. S'il y a encore un groupe, on remplace par une consonne, etc.

b. Numération en base 2

Comptez vos doigts en base 2. Quel est le lien entre l'écriture d'un nombre en base 2 et les restes des divisions successives d'un nombre par 2 ?
Proposez une procédure `bine :=proc(n)` qui renvoie la liste des chiffres composant l'écriture binaire d'un nombre. Vous utiliserez à bon escient les commandes MAPLE `iquo(a,b)` et `irem(a,b)` qui renvoient respectivement le quotient et le reste de la division euclidienne de a par b.

« Retournez une liste »
Soit `L:=[1,2,3,4,5]` une liste. Regardez ce que donne

```
M:=NULL:
for k to nops(L) do M:=M,L[-k]; od:
[M];
```

Peut-être en ferez-vous bon usage...

c. Évaluation booléenne

La commande `evalb(test)` permet de tester si une proposition est vraie ou fausse :

```
evalb(irem(8,2)=0);
evalb(irem(8,3)=0);
```

Cela nous permet de faire des tests à l'aide de `if then else` :

```
pair:=proc(n)
 if evalb(irem(n,2)=0)=true then print(‘Pair‘) else print(‘impair‘); fi;
end:

pair(45);
```

d. Rajouter un élément à la fin d'une liste

Observez les lignes suivantes :

```
L:=[1,2,3];
[op(L),4];
```

e. Coder en bibinaire

Écrivez les nombres de 0 à 25 en bibinaire. Pourquoi Boby a-t-il finement choisi le H ?
Écrivez les nombres de 255 à 257 en bibinaire. Quel est la base du bibinaire ?
Pourquoi est-il pratique d'écrire les nombres en base 2 avec un nombre de chiffres multiple de 4 avant de les traduire en bibinaire ?
Que pensez-vous de ces quelques lignes de code :

```
ib:=bine(n);
while evalb(irem(nops(ib),4)=0)=false do ib:=[(op(ib),0)];
```

À l'aide d'une série de tests `if then else` proposez une procédure `bibine:=proc(n)` qui donne l'écriture bibinaire d'un nombre entré sous forme décimale.
Testez alors :

```
bibine(2751);
```

II - Décodez du bibinaire

a. La commande table

On construit une table avec la commande... `table` :

```
note:=table([(maths)=18,(physique)=17,(info)=7]);
note[maths];
note[info];
```

À quoi correspond alors le code suivant :

```
decod:=table([seq(bibine(k)=k,k=0..15)]);
```

b. Procédure de décodage

Proposez à présent une procédure `enibib:=proc(L)` qui renvoie un nombre sous forme décimal correspondant à la liste L des lettres de son écriture bibinaire.
Vous utiliserez une seule boucle `for` connaissant la base du bibinaire.

III - Un peu d'algèbre : les groupes finis

On considèrera par la suite un groupe multiplicatif $(G,\cdot)$
Un groupe fini est tout simplement un groupe ayant un nombre fini d'éléments.
Un groupe fini est cyclique si chacun de ses éléments peut s'écrire comme une puissance d'un élément particulier g du groupe appelé générateur.
On appelle ordre de a le plus petit entier naturel non nul n vérifiant $a^n = 1_G$

a. Racines n-ièmes de l'unité

Dans la suite, on notera $\mathbb{U}_n$ l'ensemble des racines n-èmes de l'unité.
À l'aide des commandes `seq` et `exp`, donnez la *liste* des 10 racines 10-èmes de l'unité.

b. Procédure ordre

Construisez une procédure `ordre:=proc(k,U)` qui donne l'ordre du $k^{\text{ème}}$ élément de U.
Vous utiliserez `while`, `<>` qui veut dire $\neq$ et `evalc()`, cette dernière commande renvoyant la forme algébrique d'un nombre complexe.
Vous vous en servirez pour donner la liste des ordres de $\mathbb{U}_{10}$ par exemple.

c. Tables de multiplications...

Construisez les tables de multiplications (ou plutôt les tables de Pythagore) de $\mathbb{U}_3$.
Vous vous servirez intelligemment des lignes suivantes

```
A:=[seq([seq(evalc(U[i]*U[j]),j=1..n)],i=1..n)]:
T:=convert(A,array);
```

Déduisez-en que $\mathbb{U}_3$ est un groupe.
Écrivez une procédure `GroupeFini:=proc(n)` qui donnera $\mathbb{U}_n$, la liste des ordres des éléments et la table de Pythagore de $(\mathbb{U}_n,\cdot)$.
Est-ce que les $\mathbb{U}_n$ sont cycliques