Université Bordeaux 1 — **N1MA8W02 – Master 1**
Mathématiques — Année 2014–2015

Algèbre et calcul formel
Devoir Surveillé du 4 mars 2015
Durée 1h30.

À la fin de l'épreuve, votre fichier "`votre_nom.sage`" est à envoyer à l'adresse : `arnaud.jehanne@u-bordeaux.fr`

Exercice 1 – [Interpolation d'Hermite]
On cherche à trouver un polynôme P de $\mathbb{Q}[x]$ de degré inférieur ou égal à 4 tel que $P(1) = 1$, $P'(1) = 1$, $P''(1) = 2$, $P(2) = 1$ et $P'(2) = 2$.

1) Sur votre copie, traduire ces contraintes en termes de problème de restes chinois.

2) Résoudre le problème en utilisant sage, par exemple la commande `crt`. Sur votre copie, écrire la commande utilisée et le résultat obtenu.

Rappel. Pour définir $\mathbb{Q}[x]$, on peut utiliser la commande

```
Qx.<x>=PolynomialRing(QQ)
```

Exercice 2 – [Tri fusion]
Soit `l` une liste de longueur n dont les éléments sont des entiers. On veut trier cette liste. C'est-à-dire que l'on veut écrire une fonction qui en entrée prend la liste `l` et en sortie rend la liste des éléments de `l` rangés dans l'ordre croissant. Pour cela, nous utilisons l'algorithme *tri fusion*. Cet algorithme partage la liste `l` en deux listes `l1` et `l2` de tailles respectives $n_1 = \lfloor n/2 \rfloor$ et $n - n_1$. Ensuite, il se rappelle récursivement pour trier les listes `l1` et `l2`. Enfin, il fusionne les listes `l1` et `l2` pour construire la liste `l` triée.

1) Écrire une fonction `fusion` qui, étant données deux listes `l1` et `l2`, toutes deux rangées dans l'ordre croissant, rend en sortie la liste `l` des élements de `l1` et de `l2` écrits dans l'ordre croissant.

Pour cela, on compare à chaque pas `l1[i]` et `l2[j]` (en commençant avec $i = j = 0$). Si `l1[i]<l2[j]`, on place `l1[i]` dans la nouvelle liste `l` et on incrémente i. Sinon, c'est `l2[j]` que l'on place dans la nouvelle liste et c'est j que l'on incrémente. Si $i = n_1$ (resp. $j = n_2$), il n'y a pas de comparaison à faire et la liste `l` est complétée avec les éléments restants de `l2` (resp. `l1`).

2) Écrire une fonction `tri_fusion` qui trie une liste `l` suivant l'algorithme *tri fusion* rappelé ci-dessus.

Rappels. La commande `l1+l2` concatène deux listes `l1` et `l2`. Pour ajouter un élément `a` à une liste `l`, on peut taper `l.append(a)`, ou bien `l=l+[a]`.

Exercice 3 – [Algorithme d'Euclide binaire]

Cet exercice est à rédiger sur votre copie.

Algorithme 1. Euclide

Entrées: Deux entiers strictement positifs a et b
Sorties: $\text{pgcd}(a, b)$

1: Si $a = b$, alors retourner a
2: Si a et b sont pairs, alors retourner `2*Euclide(a/2,b/2)`
3: Si l'un exactement des deux entiers est pair, mettons a, alors retourner `Euclide(a/2,b)`
4: Si a et b sont impairs, et mettons que $a > b$, alors retourner `Euclide((a-b)/2,b)`

---

1) Exécuter à la main cet algorithme avec les données $a = 60$ et $b = 20$.

2) Montrer que le nombre d'appels récursifs lors de l'exécution de l'algorithme est inférieur à $\lfloor \log_2(a) \rfloor + \lfloor \log_2(b) \rfloor$. En déduire que si a et b sont inférieurs à n, la complexité binaire de l'algorithme est en $O\left((\log n)^2\right)$.

3) Expliquer pourquoi cet algorithme calcule bien $\text{pgcd}(a, b)$.

4) Modifier l'algorithme de telle sorte qu'il calcule en plus une relation de Bézout pour a et b.