# Feuille de TP n°6

**Exercice 1.** Pour ce TP, on utilisera le langage C. Soit $N$ un entier. On rappelle que pour faire des calculs modulo $N$, la méthode de Montgomery consiste à utiliser un entier $R$ premier avec $N$ et une transformation $x \mapsto xR$, et à réaliser les calculs avec les $xR$ plutôt que directement avec $x$. Notamment, par cette transformation, la multiplication s'obtient par $xy \mapsto xyR = (xR)(yR)R^{-1}$, d'où l'intérêt de savoir calculer rapidement $zR^{-1} \pmod N$ pour un entier $z$. L'algorithme de réduction de Montgomery (qui, étant donné $z$, renvoie $zR^{-1} \pmod N$) peut être donné sous la forme suivante :

---

**reduction**$(z)$

1: $q := zN^{-1} \pmod R$

2: $z' := (z - q)/R$

3: si $z' > 0$, alors renvoyer $z'$, sinon renvoyer $z' + N$

---

Pour que cet algorithme soit efficace, on choisit pour $R$ une puissance de la base (donc, en C, on prendra $R$ de la forme $2^k$), de sorte que les opérations de réduction modulo $R$ et de division ou multiplication par $R$ soient très simples et rapides. La réduction de Montgomery trouve particulièrement son intérêt dans le cas où l'on manipule de très grands entiers[1].

Voici quelques opérateurs sur les bits qui seront utiles : `&`, `|` et `^`, le *et*, le *ou* et le *ou exclusif* bit-à-bit ; `>>` et `<<` les opérateurs de décalage des bits vers la droite et vers la gauche (par exemple, `n << 3` ajoute trois 0 à droit de l'écriture en binaire de `n`).

On déclarera les variables $N$, $k$ et $R = 2^k$ en variables globales, ainsi que toutes les éventuelles variables dont le calcul ne doit être fait qu'une fois.

**1.** Programmer l'algorithme d'Euclide étendu afin de renvoyer l'inverse du premier argument modulo le deuxième.

**2.** Programmer l'algorithme de réduction de Montgomery. On pourra utiliser l'opérateur `&` pour la réduction modulo $R$ et `>>` pour effectuer la division par $R$.

**3.** Programmer un algorithme d'exponentiation rapide modulo $N$ en utilisant des réductions de Montgomery.

[1] Ceci peut se faire en C à l'aide de la bibliothèque GMP. Cependant, pour ce TP, nous nous contenterons de manipuler des `int` ou des `long` de 64 bits.