

Partiel MPSOC

Novembre 2005

Durée 2 heures

Exercice 1 – Un peu de RSA ?

Cet exercice est noté sur 10 points. Pour toutes les fonctions demandées dans cet exercice, vous donnerez leur signature. Tout au long de cet exercice, vous pourrez utiliser les opérateurs `/` et `mod`. Si `a` et `b` sont deux `int`, `a/b` renvoie le quotient de la division euclidienne de `a` par `b` et `a mod b` renvoie le reste de la division euclidienne de `a` par `b`.

Question 1 – 1 point

Écrire une fonction `est_premier` prenant en entrée un entier `n` et qui teste si celui-ci est premier. Un entier est premier si il est divisible par exactement deux nombres : 1 et lui-même.

Question 2 – 2 points

Écrire une fonction `phi` qui prend en entrée un entier `n` et retourne le nombre d'entiers dans l'intervalle $[1, \dots, n - 1]$ qui sont premiers avec `n`. On dit qu'un entier `m` est premier avec `n` si 1 est le seul entier de l'intervalle $[1, \dots, \min(n, m)]$ qui divise `n` et `m` à la fois.

Question 3 – 1 point

Écrire une fonction `presque_premier` prenant en entrée un paramètre entier `n` et qui teste si `n` est un produit de deux premiers exactement.

Question 4 – 2 points

Écrire une fonction `verif` qui prend en entrée deux entiers `n` et `m` et qui :

- lève une exception `MauvaiseEntree` si `n` et `m` sont égaux,
- lève une exception `NonPremier` si `n` ou `m` n'est pas premier,
- teste que `phi (n*m)` est bien égale à $(n-1) * (m-1)$.

Question 5 – 1 point

Écrire une fonction qui prend en entrée un entier `e` et un entier `n` (tels que `e` est inférieur à `n`) et qui teste si `e` est premier avec `n`.

Question 6 – 2 points

Écrire une fonction qui prend en entrée un entier `n` et un entier `e` et qui lève une exception `MauvaisChoix` si `e` et `n` ne sont pas premiers entre eux, et sinon, renvoie le plus petit entier `m` tel que $m * (\text{phi } n) + 1$ soit divisible par `e`.

Question 7 – 1 point

Écrire une fonction `codage` qui prend en entrée un entier `m`, un entier `n`, un entier `e` et qui renvoie $m^e \text{ mod } n$.

Vous venez d’implanter les briques de base nécessaires au crypto-système RSA.

Exercice 2 – Notation polonaise inverse

Cet exercice est noté sur 15 points.

Dans les formules mathématiques, les opérateurs binaires sont traditionnellement placés entre les deux opérandes. Par exemple, on écrit plus souvent $x + y$ plutôt que $x y +$. La forme traditionnelle est appelée *notation infixe*, celle où l’opérateur suit les opérandes est appelée *notation postfixée* ou *polonaise inversée* (en référence au mathématicien polonais Lukasiewicz qui en étudia les propriétés). On se propose ici de concevoir un programme permettant de manipuler une représentation particulière des expressions en notation infixe, et de les convertir en notation polonaise inversée.

Question 1 – 1 point

On considérera des opérateurs `Plus`, `Moins`, `Mult` et `Div` ainsi que des variables qui sont représentées par des caractères. Définir les types `operateur` et `variable`.

Question 2 – 2 point

Définir un type `element` représentant les éléments des expressions que nous allons considérer (c’est-à-dire des opérateurs ou des variables). Donner une fonction qui étant donné objet de type `element` renvoie `true` si cet élément est un opérateur.

Définir un type `exp_inf` représentant les expressions mathématiques sous la forme d’une liste d’objets de type `element`.

Question 3 – 2 points

Écrire une fonction `couper` prenant en entrée un objet `l` de type `exp_inf` (dont on suppose qu’elle est de longueur impaire supérieure ou égale à trois) et renvoyant un triplet `(l1, l2, elt)` tel que :

- `l` est la concaténation de `l1`, `elt` et `l2` ;
- `elt` est le $(N+1)/2$ -ième élément de `l` où `N` est la longueur de `l`.

Question 4 – 2 points

Une formule représentée par un objet de type `exp_inf` est bien formée si cette formule est constituée :

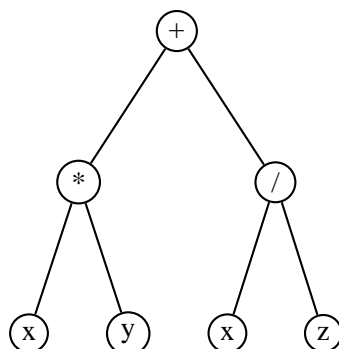
- soit d’une seule variable
- soit de la concaténation d’une formule bien formée (représentée par un objet de type `exp_inf` de longueur `N`), d’un opérateur, et d’une formule bien formée (représentée aussi par un objet de type `exp_inf` de longueur `N`).

Écrire une fonction qui teste si un objet de type `exp_inf` est bien formée en renvoyant `true` si c’est le cas et levant une exception `ExpressionMalFormee` si ce n’est pas le cas.

Des listes aux arbres. On s’intéresse maintenant à une représentation des formules en notation infixe par arbres binaires. Chaque nœud contient une *étiquette* qui est soit un opérateur, soit une variable et dont les deux fils sont eux-mêmes des arbres éventuellement vides. Si l’étiquette d’un nœud contient une variable alors c’est une feuille. Par exemple, la formule

$$((x * y) + (x / z))$$

est représentée par l’arbre :



L'algorithme effectuant la conversion d'une représentation en liste vers une représentation en arbres fait l'hypothèse que la liste passée en paramètres est bien formée et fonctionne de la manière suivante :

- Si la liste est vide on renvoie l'arbre vide.
- Si la liste est constituée de un élément, on renvoie l'arbre contenant à sa racine cet élément, le fils gauche et le fils droit sont vides.
- Si la liste contient au moins trois éléments, on note $(l1, l2, elt)$ le triplet renvoyé par l'application de `couper` sur notre liste. On renvoie alors l'arbre contenant à sa racine le `elt`, son fils gauche est le résultat renvoyé par un appel récursif de notre fonction sur `l1` et son fils droit est le résultat renvoyé par un appel récursif sur `l2`.

Question 5 – 2 points

Définir un type `etiquette` ainsi qu'un type `arbre_infixe` représentant les formules mathématiques en notation infixe bien formées sous la forme d'arbres décrits plus haut.

Question 6 – 2 points

Écrire une fonction prenant en entrée un objet de type `exp_inf` qui représente une formule en notation infixe bien formée et le convertissant en un objet de type `arbre_infixe` représentant la même formule.

Question 7 – 1 point

Modifier votre fonction précédente pour qu'elle lance l'exception `ExpressionMalFormee` dans le cas où le paramètre est une expression mal formée.

Une expression polonaise est vue comme une liste de variables et d'opérandes.

Question 8 – 1 point

Définir un type `exp_pol` pour les expressions en notation polonaise inversée.

On veut maintenant programmer une fonction prenant un objet de type `arbre_infixe` en entrée et le convertissant en un objet de type `exp_pol`. Cette fonction procède de la manière suivante :

- Si l'arbre est vide, on renvoie la liste vide,
- Sinon on concatène le résultat renvoyé par notre fonction appliqué au sous-arbre gauche de notre arbre, le résultat renvoyé par notre fonction appliqué au sous-arbre droit de notre arbre et l'étiquette de la racine de notre arbre.

Question 9 – 2 points

Donner une fonction `conversion` permettant de convertir un objet de type `exp_inf` en un objet de type `exp_pol`. Lorsque l'entrée est mal formée, l'exception `ExpressionMalFormee` devra être levée. Donner le résultat renvoyé, sur les deux formules suivantes :

$$x * y + x / z$$
$$u / v + x$$