LOGIQUE/Calculabilité
Devoir sur table (un peu simplifié)
Novembre 2006

Vous pouvez utiliser les notes de cours et documents fournis concernant la calculabilité.
Vous pouvez utiliser tous les résultats vus en cours et en exercice sans avoir à les justifier à nouveau.

Nombres premiers

Un entier n est *divible* par un entier m si $n \bmod m = 0$.

Question Donner une définition de la fonction *mod* qui calcule $n \bmod m$ et qui montre que cette fonction est primitive récursive. En déduire une fonction *dvd* telle que $dvd(n, m) = 1$ si n divise m et 0 sinon.

Un entier n est *premier* lorsque aucun d tel que $n > d > 1$ ne divise n. On pose que 0 n'est pas premier et que 1 l'est.

Question Donner une définition de la fonction *prim* telle que $prim(n) = 1$ si n est premier, 0 sinon. On peut utiliser une fonction auxiliaire à deux arguments. Justifier que le test de primalité (ie. la fonction *prim*) est primitive récursive.

On veut énumérer les nombres premiers. On commence l'énumération à 2 et on lui donne le numéro 0.

Question – en option Donner une définition de la fonction p telle que $p(k)$ soit égale au premier nombre premier supérieur à k (p doit être récursive, mais pas nécessairement primitive récursive). En déduire une définition de la fonction π telle $\pi(n)$ soit égal au n-ième nombre premier.

Boucles for

On se donne le petit langage de programmation suivant :
- symboles de constantes (pour les entiers), symboles de variables entières et symboles de fonctions.
- les expressions ont la forme d'une constante, d'une variable ou d'une application de fonction $f(e_1, \ldots, e_n)$.
- les instructions sont
  - $x := e$ avec x variable et e expression, pour l'affectation ;
  - $for\ i = 0\ to\ e\ do \ldots done$ avec i variable, e expression et ... suite d'instructions, c'est l'itération bornée usuelle pour i variant de 0 à la valeur de e ;
  - $return(e)$ avec e expression, à utiliser dans les définitions de fonction ;
  - dans une suite d'instructions, celles-ci sont séparées par des point virgules (;).
- on définit des fonctions en posant : $def\ f(x_1, \ldots, x_n) = \ldots\ endef$. Avec f symbole de fonction, $x_1, \ldots, x_n$ symboles de variables et ... une suite d'instructions entre = et $endef$. La valeur de retour est donnée par l'instruction *return*.

Question Montrer comment une fonction définie par un schéma de récurrence primitif récursif peut être définie dans ce langage.