ÉCOLE POLYTECHNIQUE – ÉCOLE NORMALE SUPÉRIEURE DE CACHAN

ÉCOLE SUPÉRIEURE DE PHYSIQUE ET DE CHIMIE INDUSTRIELLES

CONCOURS D'ADMISSION 2011 **FILIÈRE MP HORS SPÉCIALITÉ INFO**
FILIÈRE PC

COMPOSITION D'INFORMATIQUE – B – (XEC)

(Durée : 2 heures)

L'utilisation des calculatrices n'est pas autorisée pour cette épreuve.
Le langage de programmation choisi par le candidat doit être spécifié en tête de la copie.

⋆ ⋆ ⋆

Sur les permutations

La notion mathématique de permutation formalise la notion intuitive de réarrangement d'objets discernables. La permutation est une des notions fondamentales de la combinatoire, l'étude des dénombrements et des probabilités discrètes. Elle sert par exemple à étudier sudoku, Rubik's cube, etc. Plus généralement, on retrouve la notion de permutation au cœur de certaines théories des mathématiques, comme celle des groupes, des déterminants, de la symétrie, etc.

Une *permutation* est une bijection d'un ensemble E dans lui-même. On ordonne un ensemble E fini de taille n en numérotant ses éléments à partir de 1 : $x_1, x_2, \dots x_n$. En pratique, puisque seuls les réarrangements des éléments de E nous intéressent, on considère l'ensemble E_n des *indices* qui sont les entiers de 1 à n, bornes comprises. On représente alors simplement une application f sur E_n par un tableau t de taille n dont les éléments sont des indices. Autrement dit, $f(k)$ est $t[k]$, où $t[k]$ désigne le contenu de la case d'indice k du tableau t, et $t[k]$ est lui même un indice. On notera que f est une permutation, si et seulement si les contenus des cases de t sont exactement les entiers de E_n.

Dans tout le problème, les tableaux sont indicés à partir de 1. L'accès à la $i^{\text{ème}}$ case d'un tableau t de taille n est noté $t[i]$ dans l'énoncé, pour i entier compris entre 1 et n au sens large. Quel que soit le langage utilisé, on suppose que les tableaux peuvent être passés comme arguments des fonctions et renvoyés comme résultat. En outre, il existe une primitive allouer(n) pour créer un tableau de taille n (le contenu des cases du nouveau tableau ont des valeurs inconnues), et une primitive taille(t) qui renvoie la taille du tableau t. Enfin, les booléens vrai et faux sont utilisés dans certaines questions de ce problème. Bien évidemment, le candidat reste libre d'utiliser les notations propres au langage dans lequel il compose.

Partie I. Ordre d'une permutation

Question 1 Écrire une fonction estPermutation(t) qui prend une application (représentée par un tableau d'entiers t) en argument et vérifie que t représente bien une permutation. Autrement dit, la fonction renvoie vrai si t représente une permutation et faux sinon.

On suppose désormais, sans avoir à le vérifier, que les tableaux d'entiers (de taille n) donnés en arguments aux fonctions à écrire représentent bien des permutations (sur E_n). Dans cet esprit, on confond par la suite les permutations et les tableaux d'entiers qui les représentent en machine. Plus généralement, si l'énoncé contraint les arguments des fonctions à écrire, le code de ces fonction sera écrit en supposant que ces contraintes sont satisfaites.

Question 2 Écrire une fonction composer(t,u) qui prend deux permutations sur E_n en arguments et renvoie la composée $t \circ u$ de t et de u. On rappelle que la composée $f \circ g$ de deux applications est définie comme associant $f(g(x))$ à x.

Question 3 Écrire une fonction inverser(t) qui prend une permutation t en argument et renvoie la permutation inverse t^{-1}. On rappelle que l'application inverse f^{-1} d'une bijection est définie comme associant x à $f(x)$.

La notation t^k désigne t composée k fois, — la définition est correcte en raison de l'associativité de la composition. On définit l'*ordre* d'une permutation t comme le plus petit entier k non nul tel que t^k est l'identité.

Question 4 Donner un exemple de permutation d'ordre 1 et un exemple de permutation d'ordre n.

Question 5 En utilisant la fonction composer, écrire une fonction ordre(t) qui renvoie l'ordre de la permutation t.

Partie II. Manipuler les permutations

La *période* d'un indice i pour la permutation t est définie comme le plus petit entier k non nul tel que $t^k(i) = i$.

Question 6 Écrire une fonction periode(t,i) qui prend en arguments une permutation t et un indice i et qui renvoie la période de i pour t.

L'orbite de i pour la permutation t est l'ensemble des indices j tels qu'il existe k avec $t^k(i) = j$.

Question 7 Écrire une fonction estDansOrbite(t,i,j) qui prend en arguments une permutation t et deux indices, et qui renvoie vrai si j est dans l'orbite de i et faux sinon.

Une transposition est une permutation qui échange deux éléments *distincts* et laisse les autres inchangés.

Question 8 Écrire une fonction estTransposition(t) qui prend une permutation t en argument et renvoie vrai si t est une transposition et faux sinon.

Un cycle (simple) est une permutation dont exactement une des orbites est de taille strictement supérieure à un. Toutes les autres orbites, s'il y en a, sont réduites à des singletons.

Question 9 Écrire une fonction estCycle(t) qui prend une permutation t en argument et renvoie vrai si t est un cycle et faux sinon.

Partie III. Opérations efficaces sur les permutations

On commence par écrire une fonction qui calcule les périodes de tous les éléments de E_n et qui soit la plus efficace possible.

Question 10 Écrire une fonction periodes(t) qui renvoie le tableau p des périodes. C'est-à-dire que $p[i]$ est la période de l'indice i pour la permutation t. On impose un coût linéaire, c'est-à-dire que la fonction periodes effectue au plus $C \cdot n$ opérations avec C constant et n taille de t.

On envisage ensuite le calcul efficace de l'itérée t^k. On remarque en effet que $t^k(i)$ est égal à $t^r(i)$, où r est le reste de la division euclidienne de k par la période de i.

Question 11 Écrire une fonction itererEfficace(t,k) ($k \geq 0$) qui calcule l'itérée t^k en utilisant le tableau des périodes. On rappelle que t^0 est l'identité. Si besoin est, les candidats pourront utliser la primitive reste(a,b) qui renvoie le reste de la division euclidienne de a par b ($a \geq 0$, $b > 0$).

La fonction ordre de la question 5 n'est pas très efficace. En effet, elle procède à de l'ordre de o compositions de permutations, où o est l'ordre de la permutation passée en argument. Or, o est de l'ordre de $n^{\sqrt{n}}$ dans le cas le pire. On peut améliorer considérablement le calcul de l'ordre en constatant que l'ordre d'une permutation est le plus petit commun multiple des périodes des éléments.

Question 12 Donner un exemple de permutation dont l'ordre excède strictement la taille.

Question 13 Écrire une fonction pgcd(a,b) qui prend en arguments deux entiers strictement positifs a et b, et renvoie le plus grand diviseur commun de a et b. On impose un calcul efficace selon l'algorithme d'Euclide qui repose sur l'identité pgcd(a,b) = pgcd(b,r), avec r reste de la division euclidienne de a par b.

Question 14 Écrire une fonction ppcm(a,b) qui prend en arguments deux entiers strictement positifs a et b, et renvoie le plus petit commun multiple de a et b. On utilisera l'identité ppcm(a,b) = $(a \times b)$/pgdc(a,b).

Question 15 Écrire une fonction ordreEfficace(t) qui calcule l'ordre de la permutation t selon la méthode efficace. On cherchera à minimer le nombre d'appels à ppcm effectués.

* *
*