EPREUVE D'INFORMATIQUE

Durée : 3 heures

PRÉSENTATION DU SUJET

Cette épreuve comportait sept exercices, cinq de programmation, un sur la théorie des automates et un de logique.
Dans le premier exercice, le candidat devait proposer un programme pour trier une liste par dénombrement, puis effectuer une évaluation du coût dans le meilleur des cas, le pire des cas et en moyenne.
Le deuxième exercice demandait deux programmes (indépendants l'un de l'autre) : un pour remplir une matrice avec une liste donnée dans un ordre fixé (a) et un qui calculait le plus grand nombre de valeurs identiques consécutives dans un tableau (b).
Le troisième exercice était un exercice de lecture de programmes écrits dans un pseudo-langage: pour le premier et le troisième programme, le candidat devait déterminer ce que faisait le programme (le premier prenait en entrée une liste et donnait en sortie la liste obtenue en supprimant les occurrences multiples d'un élément après sa première apparition classée dans l'ordre inverse (le jury a accepté le même ordre, l'utilisation du pseudo-langage introduisant une ambigüité sur ce point), le troisième calculait le maximum des sommes de nombres positifs consécutifs dans un tableau) ; pour le second, le candidat devait détailler une exécution du programme sur une entrée particulière puis déterminer ce que faisait le programme (ce programme prenait en entrée deux nombres naturels n et p et calculait le nombre de manières de décomposer n comme somme de p entiers naturels strictement positifs).

Dans le quatrième exercice, le candidat devait rechercher une liste d'entiers naturels vérifiant une condition particulière visible sur l'écriture en base 10 de ces nombres.
Dans le cinquième exercice, on proposait comme représentation des réunions finies d'intervalles fermés disjoints la liste ordonnée des extrémités de ces intervalles. Le candidat devait proposer un programme qui déterminait si une liste ordonnée représentait ou non une réunion finie d'intervalles fermés disjoints. Puis il devait proposer des programmes pour calculer à partir de deux listes représentant des réunions finies d'intervalles fermés disjoints les listes représentant leur intersection et leur réunion ensembliste.

Dans le sixième exercice, on proposait de montrer que la suite formée par les cardinaux de l'intersection d'un langage rationnel L avec l'ensemble des mots de longueur n, n parcourant l'ensemble des entiers naturels, est une suite récurrente, dont on peut calculer la relation à partir d'un automate reconnaissant L. Deux exemples étaient détaillés, un introductif et un en application.

Le septième exercice étudiait une classe particulière de fonctions booléennes en détaillant aussi deux exemples.

ANALYSE PAR EXERCICE

Le programme de l'exercice 1 a été très souvent correctement présenté. Le jury a apprécié l'efficacité du programme (il suffit de parcourir la liste une seule fois). Les calculs de complexité sont rarement abordés ; beaucoup de candidats ne comprennent pas le sens d'un calcul de complexité et discutent la taille des données plutôt que l'efficacité du programme.
Il est exceptionnel de trouver une définition correcte de la complexité en moyenne.

L'exercice 2 a) a été souvent traité correctement. Le plus simple était d'initialiser la matrice par la matrice nulle, puis de remplacer les coefficients de la matrice par ceux de la liste tant

que c'était possible. L'exercice 2b) a été déjà moins souvent abordé et, quand il l'était, il arrivait que seules des réponses partielles soient proposées (soit seul le sous-programme pour calculer
le maximum d'une variable convenait, soit seule la variable calculant le nombre de valeurs identiques consécutives était correctement programmée).

L'exercice 3, quand il était abordé, ne proposait dans la plupart des copies que l'exécution de 3b. Même quand le résultat était juste, il était exceptionnel de trouver une justification (rôle des différentes variables, découpage logique du programme) pourtant bien utile pour arriver au résultat. Pour décrire la sortie d'un programme, il suffit d'une phrase suffisamment précise et compréhensible, ce n'était pas toujours le cas.

L'exercice 4 a été souvent correctement traité. Là encore, le jury a apprécié l'efficacité du programme proposé: il est plus facile de construire le nombre à partir de la liste des chiffres de son écriture en base 10 que de partir d'un nombre et de chercher à construire la liste des chiffres de son écriture en base 10.

L'exercice 5 était plus difficile et de nombreux candidats s'en sont tenus à la première question de vérification ; là encore, le test pouvait être effectué avec un unique parcours de la liste. Les deux questions suivantes, plus fastidieuses à écrire pouvaient se traiter avec un programme récursif. Il fallait s'assurer bien entendu qu'à chaque étape, la longueur d'une des deux listes considérées diminuait strictement, afin de garantir l'absence de bouclage et ne pas oublier de cas (le jury a apprécié les copies qui proposaient une justification rapide de l'exhaustivité des cas, par exemple avec un tableau des positions relatives des premiers intervalles de chacune des listes).

L'exercice 6 montre que la théorie des automates - au programme - est loin d'être assimilée par de nombreux candidats. Même sur des questions très simples comme le dessin d'un automate à deux états de la question 2a) ou l'expression rationnelle demandée à la question 4a), de nombreux candidats proposent des solutions erronées. Dans un tel exercice, le candidat doit argumenter ses réponses comme il pourrait le faire lors d'une épreuve de mathématiques. Les réponses doivent être précises. Par exemple, dans la question 2b où l'on demandait de démontrer que le langage considéré était la réunion disjointe de deux langages, le mot « disjoint » devait être justifié, ce qui n'était pas souvent fait.

La première question de l'exercice 7 a été dans l'ensemble réussie. Dés la deuxième question, on peut trouver dans de nombreuses copies une confusion entre la fonction booléenne considérée et sa valeur en un uplet particulier ; de plus, la question demandait une équivalence entre deux propositions ; de nombreuses copies n'ont traité que l'implication la plus facile à obtenir. La troisième pouvait se traiter de plusieurs manières ; soit comme application de la seconde question, ce qui était fastidieux mais pouvait être mené à terme correctement surtout si on savait manipuler les lois de Morgan, soit directement en séparant les cas du u-plet dont toutes les composantes valent 1 de ceux dont au moins une composante vaut 0, ce qui explicite immédiatement la fonction g.

CONSEILS AUX FUTURS CANDIDATS

Il n'est bien sûr pas facile d'écrire sur papier un programme sans pouvoir le tester. La correction est donc plus axée sur la cohérence de la démarche que sur l'exactitude absolue de la syntaxe. Le correcteur conseille aux candidats de clairement commenter leurs programmes, afin d'éviter certaines confusions. On peut lors de l'introduction d'une variable décrire en commentaire le rôle qu'elle va jouer dans le programme et justifier son

initialisation. Si le programme s'avère long, il est raisonnable de le découper en divers sous-programmes dont on explicite la finalité.
Pour la lecture d'un programme, on peut aussi expliquer le rôle de chaque variable et la finalité des sous-programmes.
Les exercices plus théoriques doivent être argumentés précisément, tout comme des exercices de mathématiques. La connaissance du cours est une nécessité.
Il n'est pas nécessaire de traiter l'intégralité des exercices pour avoir une note correcte. Le sujet est volontairement long afin d'explorer les divers aspects du programme, mais il peut être plus payant de s'attarder sur un exercice afin de le traiter correctement et précisément que de passer rapidement d'un exercice à l'autre sans apporter de réponses convaincantes.