Composition d'Informatique (2 heures), Filière PC (XEC)

Rapport de MM. Dominique ROSSIN et Etienne LOZES, correcteurs.

1. Statistiques

Rappellons que cette épreuve n'est corrigée que pour les candidats admissibles. Cette épreuve est commune entre les filières MP et PC, mais ce rapport ne concerne que les copies PC, les copies MP étant corrigées séparément. Les statistiques présentées portent donc sur le concours PC, candidats étrangers inclus.

$0 \leqslant N < 4$	58	11,8 %
$4 \leqslant N < 8$	155	31,6 %
$8 \leqslant N < 12$	184	37,6 %
$12 \leqslant N < 16$	73	14,9 %
$16 \leqslant N \leqslant 20$	20	4,1 %
Total	490	100 %
Nombre de copies : 490		
Note moyenne : 8,72		
Écart-type : 3,87		
Note minimale : 0,2		
Note maximale : 19,2		

Les candidats ont majoritairement traité cette épreuve en répondant aux questions dans l'ordre à l'exception de la question 5. Au final, 9% des candidats ont terminé cette épreuve (c'est-à-dire ont tenté de répondre aux 15 questions).

Le choix du langage de programmation s'est porté majoritairement sur Maple, comme les autres années. Ce choix s'est même affirmé et s'accompagne d'un recul de Python. Il semble important de ne pas interpréter les sujets et les rapports présents et passés comme influant le choix du langage de programmation. Ce choix est entièrement ouvert à tout langage en usage de nos jours, le choix du candidat devant se baser avant tout sur le langage avec lequel il a le plus de familiarité.

Maple	483	(92,9%)
Python	27	(5,2%)
Mathematica	9	(1,7%)
C++	1	(0,2%)
Copies blanches	6	

2. Commentaires généraux

Le sujet portait sur le problème dit du « sandwich au jambon », problème qui généralisait la notion de médian aux familles de points d'un espace de dimension n. Dans un premier temps, on proposait de se familiariser avec la représentation de telles familles (un tableau et les bornes de l'intervalle d'indices des entrées du tableau à considérer) en écrivant deux fonctions assez simples. Dans une deuxième partie, on s'intéressait au problème de la recherche d'un médian d'une famille de réels de manière efficace en se basant sur une approche « diviser pour régner ». Dans une troisième partie, on cherchait à séparer une famille de points du plan en quatre parties équilibrées délimitées par deux droites affines, et on faisait appel pour cela à la fonction médian définie précédemment.

Hormis la question 5, toutes les questions posées reposaient sur l'écriture d'un code plus ou moins long (une quinzaine de lignes pour les questions les plus difficiles) qui pouvait faire appel à des fonctions définies aux questions précédentes ou admises par l'énoncé. La question 5 demandait d'évaluer le temps de calcul @de l'un des algorithmes donné dans l'énoncé.

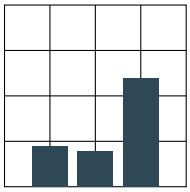
Cette année encore, une majorité de candidats montre une certaine familiarité avec le langage de programmation qu'ils choisissent. La minorité de candidats qui commet des erreurs grossières n'en apparaît que plus marginalisée. Rappellons que la correction syntaxique attendue est celle qui permet de reconstituer sans ambiguïté et avec peu d'efforts un code assimilable par un compilateur ou un interpréteur. La préparation à cette épreuve passe donc par des travaux pratiques sur machine.

Un mot sur la récursivité. Pour certaines questions, notamment cette année pour la question 4, la solution la plus naturelle fait intervenir une fonction récursive. Il est frappant que relativement peu de candidats semblent à l'aise avec ce style de programmation. Certains candidats transforment une récursivité terminale en une boucle avec succès, sans que l'on sache trop s'il s'agit d'un souci d'optimisation, ce qui serait très favorablement apprécié si c'était explicité, ou plus prosaïquement d'une ignorance de la récursivité.

3. Commentaires par question

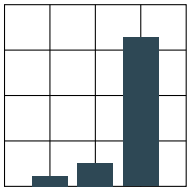
Pour donner un aperçu de la réussite par question, nous avons groupé les réponses à chaque question en trois catégories :

- faible : le candidat a reçu moins de la moitié des points
- moyen : le candidat a reçu au moins la moitié des points
- bon : le candidat a reçu la totalité des points



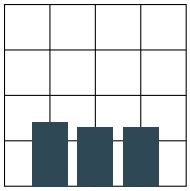
Question 1

Question 1 : Question facile mais nécessitant de bien lire l'énoncé pour ne pas oublier de prendre en compte les bornes a et b de l'intervalle de travail, ou bien renvoyer la valeur maximale et non l'un de "ces indices".



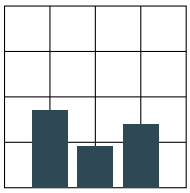
Question 2

Question 2 : Question classique et facile. Elle a été très bien traitée en grande majorité.



Question 3

Question 3 : Question relativement difficile, en partie du fait de la variété des solutions envisageables, mais aussi du fait de la suggestion plus ou moins implicite de l'énoncé de rechercher un algorithme « en place » qui permute les éléments du tableau sans allouer un nouveau tableau. Les solutions peu efficaces (allocation de tableaux auxiliaires, recopies inutiles, passes multiples) ont obtenu la note maximale du moment qu'elles étaient correctes, tandis que les solutions approximatives mais témoignant d'un souci d'optimisation ont été moins pénalisées que les autres.

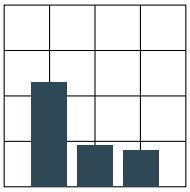


Question 4

Question 4 : Question révélatrice sur la maîtrise de la récursivité, et sur l'aptitude du candidat à comprendre un algorithme donné. De nombreux candidats n'ont pas vu la récursivité, ce qui a parfois donné lieu à du code tel que

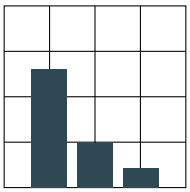
```
[...]
if i-a+1>k then tab[a..i-1][k]
[...]
```

Certains candidats ont tout de même proposé une solution itérative correcte.



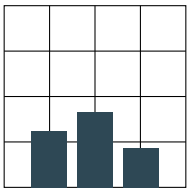
Question 5

Question 5 : Question très mal traitée et très rarement justifiée, mais notée avec clémence, en tenant compte de l'échec déjà patent à la question précédente quant à la compréhension de l'algorithme récursif.



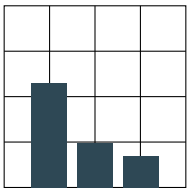
Question 6

Question 6 : Question difficile à plusieurs égards. D'une part, l'énoncé explicitait ici clairement l'aspect « en place » de l'algorithme (il fallait échanger les cases du début avec celles des pivots potentiels). D'autre part, l'algorithme était par nature récursif, et l'élimination de cette récursivité conduisait à augmenter le nombre de boucles imbriquées, ce qui pouvait perturber certains candidats. Enfin, le cas particulier du « petit » paquet résiduel demandait d'utiliser correctement la fonction floor.



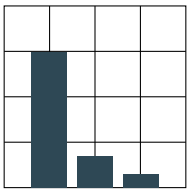
Question 7

Question 7 : Question triviale et néanmoins traitée de manière trop souvent hasardeuse, certains candidats n'ayant pas saisi le lien avec la fonction `indiceMedian` que l'énoncé demandait d'admettre.



Question 8

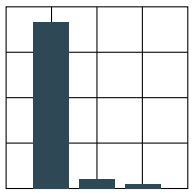
Question 8 : Question relativement facile, qui demandait de construire le tableau des angles issus des points du demi-plan supérieur et d'en calculer le médian. Certains candidats ne se sont pas restreints au demi-plan supérieur, d'autres ont renvoyé l'indice du médian et non le médian, etc.



Question 9

Question 9 : Question d'approche très similaire à la précédente, sauf qu'il valait mieux éviter d'appeler la fonction `indiceMedian`. Certains candidats ont ainsi renvoyé 0 dans le seul cas où θ était l'un des angles issus du demi-plan inférieur, donnant lieu à du code tel que

```
[...]
if angles[indiceMedian(angles,1,m)]==theta then return 0
[...]
```



Question 10

Question 10 : Question de synthèse de difficulté modérée, mais notée avec beaucoup d'exigence, qui demandait d'une part de mettre en œuvre une dichotomie, d'autre part nécessitait un peu d'attention, en particulier un incrément de π lors de la vérification de l'angle de la seconde droite.