S Informatique 1 MPI

Q1 - C'est l'écriture standard des entiers qu'il fallait donner, pas l'entier lui-même. En donnant l'écriture, une réponse de la forme

$$1\dots1 \text{ avec } c+1 \text{ uns}$$

était plus facile à lire qu'une réponse de la forme

$$\forall i \in [\![0, c]\!],\ g_i = 1.$$

Attention cependant : une réponse de la forme

$$1\dots1$$

sans préciser le nombre de uns (il pourrait y en avoir c, $c+1$, ou $c-1$) ne répond pas à la question !
Q2 - Ici, il est attendu de donner l'écriture gauche *et* l'entier. Pour prouver la réponse, une récurrence suffit, mais c'est un peu lourd. L'essentiel était surtout de remarquer que $20\dots0 > 1\dots12$.
Q3 - De nombreuses de copies contiennent beaucoup de texte mais ne prouvent rien. Des phrases comme

« Il est évident que le chiffre de plus fort poids est à la même position »

ne répondent pas à la question : c'est exactement cette « évidence » qu'il est demandé de prouver ! Certaines copies essayent d'utiliser l'unicité de l'écriture en binaire, qui n'est pas applicable ici.
Q4 - Plusieurs copies montrent l'unicité de l'écriture de M_N. Ce n'est pas ce qui est demandé : il faut mieux lire le sujet ! Dire « on itère » est une manière informelle de faire une preuve par récurrence. Ne pas expliciter la preuve par récurrence est vu comme un signe d'incompréhension. Quand une preuve par récurrence est faite, il faut écrire « preuve par récurrence » quelque part (de préférence au début).
Q5 - La question n'a pas été comprise par beaucoup de copies. Il n'était pas demandé comment calculer un entier à partir d'un objet de type `rg`, mais plutôt quels objets de type `rg` correspondent à une représentation gauche. Il a souvent manqué l'encadrement sur l'entier `position`.
Q6 - C'est la première question de programmation. De nombreuses copies écrivent des puissances de 2 de manière incorrecte (cf début du rapport de jury).
Q7 - Justifier brièvement pourquoi cet algorithme incrémente.
Q8 - Il faut faire les effets de bord, pas juste le renvoi ! Il n'y a pas de `2` dans la structure `rg`. Le test `chiffres[i] == 2` n'a pas de sens et renvoie toujours faux. Il faut mettre à jour le champ `position`.
Q11 - Cette question a posé des problèmes à de nombreuses copies. En particulier :

- La hauteur de l'arbre n'est pas la taille.
- Ne pas oublier le `+ 1`.
- L'arbre vide a une hauteur de -1

Q12 - La taille de la mémoire à allouer est `sizeof(struct noeud)`, et pas `sizeof(arb)`. L'utilisation de `arb` montre une confusion entre le type pointeur et le type structure. L'usage de la fonction `assert` n'est pas bien maîtrisée.

Q13 - La propriété « Chaque étage possède 2^i nœuds » est d'une difficulté similaire à ce qu'il est demandé de prouver. L'admettre est ici considéré comme un évitement de la question.
Q14 - L'arbre pris en entrée n'est **pas** supposé de hauteur `n`. Il faut le vérifier. Cette incompréhension a d'ailleurs été la source de nombreux programmes incorrects. Il faut bien lire le sujet !
La question 13 montre une implication, et pas une équivalence. Tester si le nombre de nœuds est bon et justifier la correction par le résultat de la question 13 dénote une confusion entre implication et équivalence.
Q16 - Il faut trouver le k-ième nœud, pas la valeur k.
Q19 - Il suffit de réutiliser les fonctions précédentes. Les recoder dénote une incompréhension du sujet. L'*accès direct* est défini dans le sujet comme étant un accès avec une complexité logarithmique, **pas** constante.
Q20 - Utiliser `lg res = malloc(...)` puis renvoyer `res` est incorrect (car mal typé). Utiliser `lg* res = malloc(...)` et renvoyer `*res` est correct mais maladroit, puisque cette allocation mémoire n'est pas utile.
Q22 - Le code est souvent trop compliqué.
Q24 - Il faut penser à utiliser la fonction `lg_trouve` définie précédemment.
Q31 - Dire « Il y a une course critique » ne prouve rien : c'est un paraphrasage de la question. Expliciter une situation où la course critique a lieu. Par exemple, énumérer l'ordre des actions réalisées par les différents fils.

↑RETOUR

T Informatique 2 MPI

Oubli régulier du chiffre 1. Plusieurs candidats n'ont pas pris le temps de la réflexion et ont répondu $10^{10^{10}}$ ou n'ont pas compris la définition de la complexité de Kolmogoroff.
Q1 - De très nombreux candidats écrivent l'algorithme récursif de l'exponentiation rapide en rappelant deux fois le calcul sur $10^{n'}$ plutôt que de stocker le résultat. Certains oublient le cas d'initialisation, où n'en font qu'un seul en 0 alors que leur écriture nécessite un cas d'initialisation aussi en 1.
Q2 - Il était attendu du candidat qu'il évoque les problèmes de dépassement d'entier et, éventuellement en fonction de leur programme, de dépassement de la capacité de la pile.
Q3 - De nombreux candidats ont proposé une bijection basée sur l'écriture en base 256. Les détails précis de cette bijection n'ont que rarement été trouvés, la justification non plus. Affirmer que cela découle de *l'écriture en base 256* est une affirmation hâtive et erronée.
Q4 - On peut noter que plusieurs candidats n'ont pas bien lu l'énoncé et ont compris $\psi(m) = \{K(\phi(n)) \geq m, n \in \mathbb{N}\}$.
Q5 - La première et la dernière partie de la fonction ont été en général bien traitées. Seuls les meilleures compositions ont traité la domination asymptotique logarithmique qui découlait du calcul du nombre de chiffres en numération de position.
Q6 - Trop de candidats ont inventé leur propre terme pour cette question. Globalement les candidats ont répondu *compilateur*, *transpileur* ou *interpréteur*. L'expression compilateur était celle attendue.
Q7 - La question nécessitait une bonne compréhension du sujet.
Q8 - La majorité des candidats ont évoqué les indentations et retours à la ligne. Beaucoup d'entre eux ont aussi évoqué le nom des variables et fonctions. Il est à noté que, contrairement à ce qu'affirme certaines compositions, le caractère récursif d'une fonction n'est pas censé rendre la lecture d'un code «impénétrable», en tout cas pas pour un candidat de MPI.
Q9 - De nombreux candidats ont fait un parcours de chaîne pour le décompte de *chaque* caractère de la chaîne, voire pour les 256 caractères ASCII, voire plusieurs fois par caractère (une fois pour chaque caractère de la chaîne). Il est attendu que les candidats soient capables de faire ce décompte en un seul parcours, et, par ailleurs, sans utiliser de tableau auxiliaire.
Q10 - Les candidats ayant traité correctement la question 13 ont traité correctement celle-ci.
Q11 - Globalement bien traitées pour les deux dernières fonctions. La première fonction, probablement hors ou à la limite du programme, n'a été traitée que par très peu de candidats.
Q12 - Souvent l'incompréhension sur le fonctionnement de la première fonction a pu mener les candidats à faire des erreurs.
Q13 - Beaucoup de candidat n'ont pas utilisé l'opérateur + ou | pour la disjonction, avec parfois des équivalents corrects et d'autre fois des syntaxes erronées.
Q14 - 92% des candidats ont traité cette question. 60% d'entre eux ont proposé un automate trop éloigné de l'automate de Glushkov, probablement à cause d'une lecture trop rapide de la question. La notion d'automate local n'est pas toujours maîtrisée. Notons que trop de candidats dessinent des transitions sans aucunes étiquettes dans l'automate.
Q15 - De nombreuses erreurs sont venues d'une mauvaise compréhension du résultat à renvoyer. Seul un préfixe devait appartenir au langage.
Q16 - Globalement les candidats ont traité cette question avec de bonnes intuitions et une bonne démarche. Un candidat sur six l'ayant traité a réussi à écrire la preuve du début à la fin avec rigueur.

Q17 - La question 22 est fausse. 63 candidats ont détecté cette erreur et l'ont explicitement signifié sur leur copie, ce qui a été valorisé.
Q18 - La question 23 était difficile, mais réussie par ceux qui ont vu la coquille de la question 22.
Q19 - Globalement les candidats ont répondu par la négative à la question, en reconnaissant une variante du langage $a^n b^n$ et en proposant d'utiliser le lemme de l'étoile. Toutefois, le lemme de l'étoile a été énoncé correctement par trop peu de candidat. De plus, de nombreux candidats l'ont ensuite mal utilisé, choisissant la décomposition à leur guise, manipulant la quantification existentielle comme une universelle.
Q20 - Pour représenter un ensembles, beaucoup de candidats ont proposé un *tas binaire* ou une structure *union-find*, sans en expliciter l'implémentation. C'est l'opération de recherche qui devait ici être efficace. 40% des candidats ayant traité la question ont parlé d'ABR et 14% ont mentionné les ABR équilibrés. Les correcteurs sont surpris de ne pas avoir des pourcentages plus élevés de bonne réponse.
Q21 - La difficulté résidait dans la création de noms uniques pour chaque variable et leur bonne association.
Q22 - Peu traitée. La preuve demandait beaucoup de rigueur. Parmi les candidats ayant traité la question, beaucoup de candidats proposent une *induction sur le codage*. De plus le traitement du cas de l'opérateur d'arité deux dans le codage préfixe est presque systématiquement mal exécuté. Les candidats sautant à la conclusion sans vérifier si l'hypothèse de récurrence peut vraiment s'appliquer.

↑RETOUR