Q8 - Cette question était assez difficile. Il fallait dans un premier temps bien comprendre le type d'un point de passage (beaucoup de candidats l'ont interprété à tort comme une liste de deux flottants, contredisant l'énoncé). La conversion (en radians pour les angles, en mètres pour les longueurs) était ensuite attendue. La prise en compte de l'altitude moyenne de l'arc a porté à confusion, ainsi que l'utilisation du théorème de Pythagore. Certains candidats ont illustré leur code d'un dessin, les aidant vraisemblablement à mieux comprendre la question.

Q9 - Question plutôt réussie, excepté l'erreur fréquente sur le type d'un point de passage.

Q10 - Beaucoup trop de candidats ont cherché à utiliser une syntaxe "à la `numpy`" dans cette question, ce qui n'est pas possible avec des listes.

Q11 - Outre la difficulté de projeter l'équation différentielle sur les deux axes, de nombreux candidats ont eu des difficultés à coder correctement la force $\vec{f_B}$. Les fonctions `uniform` ou `gauss` génèrent des valeurs différentes à chaque utilisation, ce qui a rendu faux le code de beaucoup de candidats.

Q12 - Le principe de la méthode d'Euler n'est pas toujours maîtrisé. L'enchaînement des questions incitait également à utiliser les fonctions précédentes (`vma` et `derive`).

Q13 - Le principe a globalement été compris. Les erreurs ont souvent porté sur les voisins possibles de `p` (les voisins en diagonale n'étaient pas à considérer) ainsi que sur le test de leur appartenance à atteints. Les conditionnelles utilisant `elif` ne permettaient pas de répondre correctement. La construction de la liste voulue à partir d'une liste vide [ ] et de `append` successifs est un incontournable à maîtriser.

Q14 - De nombreux candidats ont utilisé un point en trop dans leur exemple de CAE, ou oublié la moitié des cas dans le dénombrement des CAE les plus courts.

Q15 - Beaucoup de candidats ont confondu la liste vide [ ] avec la valeur spéciale `None`. Trop de copies ont également recalculé deux ou trois fois la même valeur de $\texttt{positions}_\texttt{p}\texttt{ossibles}(\texttt{p}, \texttt{atteints})$ au lieu de la stocker.

Q16 - Comme souvent, la détermination rigoureuse de la complexité (dans le cas le pire) d'une fonction a beaucoup posé problème. Trop de candidats pensent qu'une telle complexité est toujours en $O(n^k)$, où k est le nombre de boucles for présentes dans la fonction. On attendait ici que l'on détermine et somme les complexités de chaque fonction appelée dans la boucle.

Q17 - L'interprétation rigoureuse de la valeur de l'ordonnée représentée dans ce graphique a souvent posé problème. Il ne s'agissait par exemple pas d'un pourcentage.

Q18 - Les tris, ainsi que leur complexité respective, ne sont pas toujours maîtrisés. Rappelons que le tri rapide a une complexité dans le cas le pire en $O(n^2)$, alors que celle du tri-fusion est en $O(n \log n)$.

Q19 - Le principe de trier la liste `chemin` a été plutôt bien compris, et la question assez réussie.

Q20 - L'obtention des formules pour les coordonnées de l'image d'un point par une rotation a souvent été entachée d'erreur.

Q21 à 23 - Figurant en fin d'énoncé, ces questions ont permis de valoriser la prise de recul et la maîtrise des listes des meilleurs candidats.

4.2 Informatique option MP

4.2.1 Généralités

Le sujet s'intéresse à l'analyse et à la programmation du jeu du solitaire. Il est composé de deux parties : une première partie avec un jeu sur le tablier européen en un minimum de coups et une autre partie où le jeu se déroule sur un tablier unidimensionnel.
Le sujet est composé de 30 questions. Elles permettent de reconnaitre des motifs, de définir des coups simples puis composés, d'identifier des parties minimales, de reconnaître des motifs sur un tablier

unidimensionnel. Les candidats ont bien saisi les différences entre le jeu sur tabliers européen et unidimensionnel. En conséquence, ils ont su gérer le passage d'une question à une autre et d'un tablier à un autre en vue de répondre au maximum de questions.
Les candidats ont abordé les questions de programmation ainsi que les questions portant sur des démonstrations. Ils ont bien compris le sujet ainsi que les différences qui résultent de l'utilisation des deux tabliers.
Comme l'année dernière, les correcteurs ont noté, avec satisfaction, qu'il y avait de nouveau une diminution importante de compositions traitant exclusivement les questions portant sur l'écriture de programmes ou les questions portant sur des démonstrations. Par contre, il reste encore quelques compositions qui utilisent un langage autre que Ocaml (Python en particulier) alors que cela est proscrit de manière explicite en préambule du sujet (page 1 du sujet).
Enfin, même si cela n'a pas porté préjudice aux candidats dans la correction, les correcteurs notent une tendance encore marquée consistant à écrire des programmes Ocaml en style impératif. Une part importante de candidats ne semblent pas entièrement à l'aise avec la programmation fonctionnelle. Les correcteurs ne privilégient pas un style par rapport à un autre mais encouragent les candidats à maitriser les deux styles de programmation afin de choisir le mieux adapté à la réponse à une question.

4.2.2 Analyse de Forme

Les programmes présentés par les candidats respectent les règles d'indentation avec des retours à la ligne facilitant la lecture du code. Plusieurs copies restent malgré tout mal présentées voire même illisibles pour le correcteur, sans indentation, avec de grosses ratures, des renvois avec des flèches en bas de page, etc. Ces copies sont très difficiles à interpréter.
Plusieurs compositions utilisent des fonctions auxiliaires, ce qui est une bonne chose pour décomposer un programme. Mais, ces fonctions portent des noms non significatifs comme `Aux`, `Aux1`, `Aux2`, etc. Lorsque ces fonctions sont ensuite appelées/composées dans le programme demandé, il est difficile de comprendre la fonction réalisée par ce programme. Il est recommandé d'utiliser des noms significatifs à l'image de ce que font les concepteurs des sujets. Un commentaire additionnel permet de mieux évaluer la compréhension du sujet ou de la question par les candidats.
Les correcteurs souhaitent mentionner la présence de preuves d'équivalence souvent imprécises lorsqu'elles sont conduites par une succession d'équivalences. Peu de compositions font la démonstration dans les deux sens.
Enfin, plusieurs copies renferment encore des expressions de la forme « C'est évident », « il est clair que ... », « Trivial ». Ces expressions ne sont pas acceptées lorsque la question exige une justification et/ou bien une démonstration qui doit être convaincante et rigoureusement établies.

4.2.3 Analyse par question

Q1 - De nombreux de candidats ignorent la distinction entre parcours en profondeur ou largeur et l'emploi associé d'une pile ou d'une file.

Q2 - Le test `x <= 6` est parfois oublié. Pour certains candidats, les correcteurs ont observé une confusion entre Python et Ocaml dans le traitement de cette question.

Q3 - L 'opération `x :: l` est interprétée comme un effet de bord sur `l`. Ce n'est pas le cas.

Q4 - Question globalement bien traitée.

Q5 - Certains candidats n'ont pas compris les spécificités du sujet et utilisent les puissances de 2 pour faire le décalage (cela n'a pas été sanctionné). Ces candidats n'ont pas perçu l'intérêt de l'approche.

Q6 - Du code plus ou moins compliqué. Beaucoup utilisent une programmation impérative plutôt que fonctionnelle, et donc avec des programmes et codes plus conséquents à écrire. Par ailleurs, on trouve également l'utilisation d'une somme plutôt qu'une opération logique bit-à-bit.

Q7 - Question globalement bien traitée.

Q8 - Certains candidats n'ont pas bien compris ou pas bien lu la définition des opérateurs logiques sur les entiers donnés dans l'énoncé. Les correcteurs relèvent des erreurs liées à des incompréhensions de l'effet du décrément sur la représentation binaire.

Q9 - Des expressions parfois très compliquées sont données en réponse à cette question.

Q10 - Le décalage est en général bien fait mais certains candidats oublient de tester si l'encoche fait toujours partie du tablier européen. Des confusions entre la valeur et la position (entre le motif $p = 2^k$ et son numéro k).

Q11 - On trouve du code parfois très compliqué pour résoudre cette question. Souvent les candidats utilisent une programmation impérative. Beaucoup de candidats oublient de vérifier certaines conditions pour que le saut soit possible et de tester si le fichet dans sa position finale est toujours dans le damier européen. De nombreux candidats dupliquent le même code ou quasiment pour tester les quatre directions.

Q12 - Les copies présentent souvent des codes confus. En particulier, nous notons des oublis de mémoriser les coups intermédiaires ou bien le renvoi du seul premier coup simple. Les candidats ayant utilisé une programmation fonctionnelle ont de meilleures réponses que ceux qui utilisent une programmation à l'aide de boucles. Les programmes qui ne géraient pas les doublons n'ont pas été sanctionnés.

Q13 - Certains candidats oublient de tester les fichets présents sur le damier européen, et peu de candidats ont le réflexe de récupérer la première composante comme indiqué par l'énoncé. Enfin, les correcteurs relèvent ici aussi la présence de confusion entre numéro k et motif $p = 2^k$.

Q14 - En général cette question est assez bien traitée par les candidats qui ont traité cette question.

Q15 - Parfois la notion d'accumulateur pour récupérer les données en sortie n'est pas traitée par les candidats qui ne font pas référence à la question précédente. D'autres candidats s'efforcent de redéfinir la question précédente sous une autre forme. Plusieurs candidats omettent d'effectuer les mouvements sur les motifs de `L`. Cette question est, en général, mieux traitée en programmation de style impératif que fonctionnel.

Q16 - Question peu traitée. Elle semble difficile pour de nombreux candidats. Certains oublient la création du dictionnaire lorsqu'ils font le choix de cette option pour traiter la question.

Q17 - Peu de candidats traitent correctement cette question. Certains oublient les solutions justes ou à peu près justes qu'ils proposent dans les questions précédentes pour ne s'intéresser qu'à la réponse fausse proposée en question 1.

Q18 - Question globalement bien traitée.

Q19 - Les compositions présentes souvent des automates très compliqués, et certaines réponses sont fausses.

Q20 - Une majorité de candidats ne connaissent pas la notion de langage. La relation avec un automate local semble confuse.

Q21 - Peu de candidats donnent la bonne réponse. On note beaucoup d'erreurs de calculs dans les majorations.

Q22 - De bonnes réponses sont données lorsque les candidats ont traité les différents cas. Autrement, on trouve des calculs parfois très confus avec beaucoup d'erreurs.

Q23 - Certains candidats ont traité partiellement ou complètement cette question dans la question précédente. Les correcteurs ont bien pris en compte ce résultat. Néanmoins, peu de candidats aboutissent à l'expression recherchée.

Q24 - Très peu de candidats fournissent la bonne réponse. Nous avons observé des réponses correspondant à de nombreuses estimations sans fondement.

Q25 - Peu de candidats analysent correctement le processus de la trace. Certains se limitent à réécrire ce qui est donné dans l'énoncé pour cette question.

Q26 - Très peu de candidats arrivent à donner, même grossièrement, le nombre fini de valeurs distinctes.

Les questions **27** à **30** sont très peu abordées.