PC

CONCOURS COMMUN INP

RAPPORT DE L'ÉPREUVE ÉCRITE D'INFORMATIQUE

1/ REMARQUES GÉNÉRALES

L'épreuve d'informatique du concours CCINP 2024 proposait de travailler sur le jeu de l'Awalé.

La première partie traitait de la compréhension des règles du jeu. La deuxième partie traitait de la programmation de la structure du jeu dans le cadre d'un jeu entre deux humains. La troisième partie traitait de la programmation d'une intelligence artificielle pour jouer à la place d'un humain.

Le sujet permettait de balayer un large ensemble des compétences décrites dans le nouveau programme d'informatique en CPGE.

Cette session a poursuivi la mise en place d'un document réponse. Les candidats l'ont globalement bien utilisé. Il est rappelé qu'il est utile de réfléchir sur un brouillon afin de porter le plus proprement possible la solution sur le document.

La propreté des copies s'est dégradée par rapport aux années précédentes. La qualité de la présentation et de la rédaction est prise en compte dans la notation.

Pour la lisibilité des programmes, il est vivement conseillé de choisir des noms de variables intelligibles, de bien soigner son indentation avec l'alignement sur les carreaux ou un trait vertical. En revanche, on évitera absolument d'utiliser un caractère censé représenter l'indentation et répété sur toute la ligne, que ce soit un trait horizontal, un point ou le caractère ␣. La lecture d'un tel code par le correcteur est très difficile et ne peut que désavantager le candidat.

L'utilisation des commentaires, bien qu'appréciée, ne doit pas surcharger la copie et faire perdre trop de temps aux candidats. Certains bons candidats ont énormément commenté leur code mais cela a eu pour conséquence de ne pas aller très loin dans le sujet. Il était écrit, avec un exemple, de ne pas recopier les signatures des fonctions qui sont présentes pour aider le candidat à comprendre le type des données d'entrées et sorties. Pourtant un certain nombres de candidats les ont recopiées et même certains ont utilisé des syntaxes comme `plateau :[int][i]`

Cette session semble marquer une nette régression quant à la maîtrise de Python. Un nombre assez important de copies ne traitent aucune question correcte en Python et prennent des points sur les questions SQL. Les correcteurs rappellent aux candidats que seule une pratique régulière du langage sur machine peut leur permettre de mettre en place un certain nombre d'automatismes dont l'absence est flagrante dans de nombreuses copies. Il est utile d'essayer de se réentraîner sur quelques anciens TP sur machine lors de la préparation aux écrits pour que ces automatismes reviennent juste avant les épreuves. En outre, s'obliger à écrire systématiquement le code sur feuille pendant toute l'année avant de

le taper à l'ordinateur pendant les TP vous permettra aussi de soigner la présentation et la lisibilité de celui-ci.

De manière non exhaustive, voici quelques erreurs très souvent relevées :

- `range 2` ou `range([0 :10])`
- `If, While, Else`, etc avec une majuscule
- Non parenthésage des conditions comportant des `and` et `or` rendant la condition fausse
- Utilisation de la division entière à la place du modulo
- Condition d'égalité avec un seul symbole =
- `Valeur = variable`, par exemple :

  `ramasser_graines(plateau, case) = graines`
- Structure : `valeur if condition` sans le `else` !
- Condition `valeur == 2 or 3`
- Appel à une fonction sans mettre les arguments `if test_case :`
- Appartenance à un ensemble de valeurs avec des syntaxes du type

  `variable in [1:6]`
- Confusion entre `` `print` `` et `` `return` ``

2/ ANALYSE DÉTAILLÉE DES QUESTIONS

Q1 : Cette question a été assez bien traitée.

Q2 : Dès cette question, on note beaucoup de candidats qui ont du mal à comprendre les règles du jeu.

Q3 : Cette autre question de compréhension permettait d'illustrer la mise en famine (ou non) de l'adversaire.

Q4 : La majorité des candidats trouve la bonne parité. En revanche, l'utilisation du modulo et d'une première condition est déjà discriminante.

L'utilisation de la syntaxe `return condition` plutôt que `if condition == True : return True else : return False` permet de montrer une bonne compréhension de la nature des booléens et des conditionnelles.

Q5 : Cette question a été moyennement bien traitée. Beaucoup de candidats n'arrivent pas à utiliser une variable intermédiaire ou à utiliser l'affectation multiple.

Q6 : Cette question a été assez mal traitée. Combien de bits pour coder l'entier 48 ? 2^{48} ? Rarement 6 malheureusement.

Q7 : Cette question a été rarement entièrement bien traitée. Il y a souvent des erreurs dans la copie des deux listes de la structure jeu. Un certain nombre de candidats ne semblent pas connaître la notion de dictionnaire, mais arrivent toutefois à les utiliser dans les questions suivantes.

Q8 : Cette question a été assez mal traitée. Seule la moitié des candidats utilise un `while` pour distribuer les graines tant qu'il en reste. Peu pensent à ne pas remplir la case d'origine où étaient les graines ou

encore à mettre à 0 cette case d'origine. Il était également important de gérer les indices des cases dans les cas où le nombre de graines à distribuer nécessitait plusieurs tours de plateau.

Q9 : Cette question a été étonnamment assez mal traitée alors qu'il suffisait de traduire en Python les conditions données dans la question. On retrouve ici beaucoup d'oublis de parenthèses rendant la condition fausse.

Q10 : Cette question a été assez mal traitée. Un certain nombre de candidats arrivent à écrire une structure de fonction récursive contenant un cas de base et un appel récursif (parfois avec le bon argument…). En revanche, assez peu de candidats arrivent à gérer les retours intermédiaires et utilisent une variable locale, qui est remise à 0 à chaque appel, pensant que celle-ci s'incrémente. En outre, les réponses correctes mais non récursives ne rapportaient aucun point étant donné qu'il était clairement indiqué qu'une réponse récursive était attendue. En revanche, les efforts même incomplets en ce sens ont été valorisés.

Q11 : Cette question est mal traitée. Les candidats n'ont pas compris qu'il fallait jouer virtuellement le coup sur une copie de la structure jeu pour déterminer si le coup affame l'adversaire ou non.

Q12 : Cette question est assez mal traitée par les candidats. Même remarque sur les conditions qu'à la Q9. La structure `True if …. else` a été rarement bien utilisée et notamment il y a souvent une absence du `else` ! On attendait néanmoins un simple `` `test = (…) and (…) and condition3` ``

Q13 : Question plutôt bien traitée.

Q14 : Cette question est assez bien traitée. Seuls les problèmes de syntaxes pénalisent les candidats. On peut bien sûr utiliser une structure en `` `if/elif/elif.../else` ``, mais il faut garder à l'esprit qu'on ne peut pas faire un "pause" pour faire un calcul puis reprendre le `` `elif` `` comme si c'était la suite de la structure précédente.

Q15 : Question assez mal traitée. Les erreurs les plus fréquentes sont que les fonctions sont appelées sans penser à stocker le retour dans une variable. Il était également important de noter que les variables étaient réutilisées dans les lignes de codes suivantes. On remarque également un nombre important de copies qui appellent `tour_joueur1` sans préciser l'argument.

Q16 : Question assez bien traitée, cependant un nombre important de candidats ne pensent pas à affecter les graines du plateau au bon joueur.

Q17 : Question assez bien traitée.

Q18 : Question moyennement bien traitée. Le graphe est souvent bien rempli, mais le coup à jouer n'est souvent pas le bon.

Q19 : Même remarque que pour la question Q15 où il faut bien gérer les appels et retours des fonctions gain et NegaAwale.

Q20 : Question classique mais rarement traitée. Il s'agissait de la recherche d'un maximum ou de l'indice du maximum d'une liste de liste.

Q21 : Question rarement traitée.

Q22/25 : Les questions de SQL sont traitées sur la majorité des copies. Les correcteurs encouragent les candidats à bien respecter l'ordre des mots clés. Il est néanmoins à noter que les syntaxes de SQL et de Python sont assez dissemblables. Il faut éviter d'utiliser des éléments syntaxiques de l'un pour essayer de coller à l'autre. En particulier des `` `jeu[0] = 'a'` `` ne peuvent pas remplacer le `` `jeu LIKE 'a%'` `` décrit dans l'énoncé.