EPREUVE ORALE D'INFORMATIQUE

1. Remarques générales

Depuis cette session 2015, le concours G2E propose une épreuve orale d'informatique, au choix avec la chimie. L'épreuve orale d'informatique dure 50 minutes : 25 minutes de préparation suivies de 25 minutes d'exposé devant l'examinateur.

Pendant la 1ère partie de l'exposé (entre 10 et 15 minutes), le candidat est amené à présenter la résolution d'un exercice tiré au sort et préparé pendant les 25 minutes préalables. Pendant la seconde moitié de l'exposé, il pouvait au choix présenter un projet préparé tout au long de son année en classe préparatoire, ou travailler sur un exercice non préparé proposé par l'examinateur.

- ✓ L'objectif des exercices proposés est de vérifier la capacité du candidat à pouvoir transformer un problème élémentaire en un algorithme et à identifier les fonctions et types de structures nécessaires à sa programmation. L'algorithme est présenté dans un langage de programmation, Python dans la majorité des cas.

  Les exercices se présentent sous forme de problèmes généraux ne faisant pas appel nécessairement à des notions mathématiques, physiques ou biologiques…

  Pendant l'exposé, le candidat est convié à présenter une solution pour résoudre le problème posé et à répondre à des questions qui peuvent être liées à la solution exposée, prolongements, variantes, efficacité de l'algorithme proposé… Les interventions de l'examinateur sont destinées à obtenir des précisions, corriger des erreurs ou de mauvaises démarches, elles ne sont jamais faites pour perturber le candidat.

  L'évaluation tient compte d'aspects strictement informatiques :
  - exactitude de l'algorithme présenté,
  - maîtrise des concepts de programmation manipulés,
  - efficacité du programme, prise en compte des cas particuliers.

  Et, plus généralement, d'autres qualités ont aussi été appréciées :
  - vivacité et rapidité suite aux remarques de l'examinateur,
  - aptitude à défendre les solutions proposées,
  - capacité à relier le problème à des problèmes plus généraux.

- ✓ Dans la 2ème partie de l'exposé, le candidat présente un projet travaillé pendant l'année scolaire. Les candidats ont pu s'appuyer sur une présentation projetée sur leur ordinateur ou imprimée. Certains n'ont utilisé aucun support hormis le script du programme implémenté.

  L'exposé sur le projet a notamment pour objectif de mettre en évidence la capacité du candidat à présenter clairement :
  - le sujet sur lequel il a travaillé
  - les hypothèses et limites fixées pour sa résolution
  - l'analyse effectuée et les solutions algorithmiques mises en œuvre pour le réaliser
  - les difficultés rencontrées et les perspectives pouvant être envisagées.

  L'évaluation tient compte de :
  - la qualité et la clarté de la présentation
  - de l'ampleur du projet : difficulté du sujet, recherche bibliographique, nombre de méthodes implémentées, analyse des résultats …
  - d'une estimation de l'investissement apporté sur le projet (nombre de participants au projet, durée sur l'année, nombre de lignes de code …)
  - de la qualité du code.

Il apparaît que l'ensemble des candidats ont choisi cette option en connaissance de cause, et à part quelques exceptions, ils ont les compétences permettant de résoudre les exercices, ce qui donne une moyenne de 14,5 à l'épreuve. Un certain nombre de candidats montrent une très bonne maîtrise des concepts manipulés et une grande aisance à écrire un algorithme. Les examinateurs tiennent à souligner que même si certains candidats ont parfois été décontenancés par le sujet et n'ont pas trouvé forcément la bonne solution au départ, les interrogateurs ont tout de même pu évaluer leur capacité à rebondir aux remarques, leur réactivité pour rectifier le tir et proposer une solution au problème posé et leurs compétences en programmation.

La palette des projets présentés a été très variée et les sujets étaient intéressants, même si on retrouve souvent les mêmes (épidémie, évolution de population, manipulation d'ADN-ARN, jeu, etc.). Nous avons cependant pu constater une grande différence au niveau du temps consacré au projet au cours de l'année et des conditions de réalisation (nombre d'élèves impliqués, recherche biblio nécessaire, nombre de méthodes implémentées, interface graphique fournie ou non, etc.), et cela se traduit par de grosses différences dans le volume et la complexité du code présenté. Il n'a donc pas toujours été évident de juger correctement du travail présenté par le candidat. Cela s'avère particulièrement vrai lorsque le projet s'appuie presqu'exclusivement sur une interface graphique, qui, même si cela nécessite un investissement certain pour comprendre les concepts manipulés, ne sollicite pas vraiment de compétences en algorithmique.

2. Quelques points d'amélioration attendus

- Il est indispensable que le candidat présente le sujet de l'exercice dans son ensemble avant de rentrer dans le détail sans aucune introduction.
- De la même façon, avant de rentrer dans le détail, chaque question doit être introduite en présentant les résultats attendus, les données fournies et brièvement la méthode mise en œuvre.
- Il faut que les candidats prennent le temps de bien lire l'énoncé et de se poser les bonnes questions avant de se lancer dans sa résolution.
- Un des très gros défauts rencontrés consiste à utiliser des noms de variables absolument peu explicites tels que A, B, C ou x, y, z. Ce défaut se retrouve également dans les scripts des projets exposés, ce qui ne favorise pas une compréhension aisée et rapide des codes présentés.
- Au niveau programmation, quelques améliorations peuvent être apportées :
  - dans l'utilisation des instructions conditionnelles, en exploitant mieux la combinaison des *if … else* ou *elif*
  - en évitant l'appel répété d'une même fonction avec les mêmes arguments. Appeler la fonction une seule fois et stocker son résultat pour le réutiliser ultérieurement
  - privilégier l'utilisation de la méthode *append* plutôt que celle de l'opérateur + quand on veut ajouter un élément dans une liste. C'est beaucoup plus efficace.
  - on peut également faciliter l'écriture de certains programmes en utilisant l'instruction *break*
  - même si dans l'ensemble les candidats maîtrisent assez bien l'instruction *return*, celle-ci est parfois mal utilisée dans une structure conditionnelle au sein d'une itération, erreur classique :

```
def fct(…):
""" doit retourner False quand la condition est vérifiée
                et True dans les autres cas"""
     for … :
          if condition:
             return False
          else:
             return True
```

  - Lorsqu'on demande de calculer un minimum ou un maximum, un certain nombre de candidats optent systématiquement pour un tri ou la construction d'une liste intermédiaire avec utilisation de la fonction *min* ou *max* de Python. C'est la plupart du temps une méthode trop coûteuse, il faut savoir rechercher un minimum ou un maximum avec un simple parcours d'une liste, ce qui permet par exemple de faire la somme des éléments en même temps.
- Les candidats semblent peu à l'aise avec les chaînes de caractères et ont parfois été un peu perturbés par les exercices les mettant en œuvre. Dans ce cas, ils ont souvent manipulé les objets impliqués comme des listes. De façon anecdotique, les constantes caractères sont souvent utilisées sans les apostrophes : A à la place de 'A' par exemple.
- Peu de candidats connaissent l'opérateur modulo "%" qui rend pourtant de nombreux services, tester si un nombre est pair par exemple…
- Peu savent utiliser également le *slicing* (découpage) de Python permettant d'extraire des sous-chaînes ou des sous-listes très facilement et rapidement.
- Lors de la présentation du projet, les candidats se sont parfois perdus dans le détail, en particulier sur les instructions d'affichage graphique.