Chapitre 7

Épreuves orales de l'option informatique

7.1 Remarques sur l'épreuve de leçon de mathématiques

Dans tous les cas, le jury a aussi interrogé le candidat sur la partie du programme non couverte par le sujet : question d'analyse pour les candidats ayant traité un sujet d'algèbre et vice-versa.

Les remarques concernant cette épreuve ne sont pas différentes des remarques concernant les épreuves de leçon des autres options, et le lecteur est invité à se reporter à la section du rapport consacrée à ce point.

Cette épreuve va évoluer à partir de l'année prochaine : le candidat tirera un couple de sujets au sein d'une liste d'une quarantaine de sujets d'algèbre et d'analyse extraite de la liste générale des autres options. **Mais il n'y aura donc plus nécessairement un sujet d'algèbre et un sujet d'analyse !** Il pourra y avoir deux sujets d'algèbre ou deux sujets d'analyse, par exemple : *Loi binomiale* et *Fonctions monotones*. Le nouveau programme précise en effet :

> *Les candidats se verront proposer deux sujets, dans un corpus d'algèbre, de géométrie, d'analyse et de probabilités.*

Il est donc impératif que les candidats ajustent leur préparation à cette nouvelle organisation. La liste des leçons sera disponible dès la rentrée sur le site Web de l'Agrégation.

7.2 Remarques détaillées sur l'épreuve de leçon d'informatique

L'épreuve de leçon d'informatique a été organisée exactement sur le modèle de celles de mathématiques. Un ensemble de 27 titres de leçons a été proposé aux candidats.

Le candidat tire un couplage de deux titres et choisit l'un des deux. Après 3 heures de préparation, il expose son plan de leçon au jury pendant une petite dizaine de minutes, ainsi que deux propositions de développement. Le jury choisit l'une des propositions, le candidat expose ce développement pendant une quinzaine de minutes, puis une discussion libre s'engage avec le jury pendant le reste du temps.

Cette épreuve va évoluer à partir de l'année prochaine : on trouvera la nouvelle liste de leçons dans le rapport 2008 qui sera disponible dès la rentrée sur le site Web de l'Agrégation. Essentiellement, les leçons de *programmation* (langages typés, sémantique, typage, compilation) ont été retirées de la liste ainsi que certaines leçons élémentaires d'*algorithmique*. Par contre, la liste de leçons a été raffinée en ce qui concerne la *logique*, en particulier les preuves de programme. Nous espérons que ces modifications contribueront à faciliter la tâche des préparateurs.

De manière générale, le jury a plutôt été heureusement surpris par la qualité de certaines leçons présentées, notamment parmi les leçons les plus avancées, ce qui confirme le bon travail des préparations spécifiques

en amont du concours. Ceci est particulièrement net dans la bonne focalisation des présentations. Beaucoup de candidats cernent bien le sujet de leurs leçons et proposent des développements intéressants mais le niveau est assez hétérogène, ce qui conduit à une grande dispersion des notes.

7.2.1 Organisation de la leçon

De même, l'organisation de la leçon est le plus souvent pertinente. On remarque cependant quelques candidats qui se présentent devant le jury avec des plans d'une seule page, peut-être par ignorance des modalités de l'épreuve.

Il est recommandé aux candidats de bien situer le domaine de la leçon avant de se lancer dans les raffinements techniques. C'est particulièrement important à cause de la largeur thématique du programme.

Une tentation bien compréhensible pour les candidats est de *mathématiser* les sujets de leçons en oubliant l'aspect informatique. Ainsi, sur le sujet *Langages algébriques*, il était tentant de faire une leçon contrée sur un aspect théorique unique comme le Lemme d'itération d'Ogden, en oubliant complètement les aspects plus concrets de ce domaine et ses multiples applications, par exemple à l'analyse syntaxique et aux compilateurs.

Le jury tient donc à rappeler qu'il s'agit bien d'une épreuve d'*informatique fondamentale*, et non pas d'outils mathématiques pour l'informatique. Il appartient au candidat de montrer la pertinence des outils mathématiques qu'il développe vis-à-vis des objectifs du thème informatique développé dans la leçon.

La présentation d'outils mathématiques pour eux-mêmes, en particulier lorsqu'il s'agit d'outils sophistiqués comme ceux de la théorie de la calculabilité ou de la théorie des types, s'apparente donc à un *hors-sujet*. Ce point avait déjà été souligné dans le rapport de l'an passé et les titres des leçons ont été affinés en conséquence. Les titres des leçons concernant des modèles formels de l'informatique sont maintenant libellés en mentionnant explicitement *exemples et applications*.

Les candidats de niveau moyen ont souvent montré des connaissances assez solides pour les résultats théoriques, mais par contre un manque de réflexion manifeste en ce qui concerne leurs applications et exemples concrets de mise en oeuvre.

Les deux questions-clés de cette épreuve sont toujours les mêmes.
– À quoi cet outil mathématique sert-il dans le cadre informatique considéré ? Pouvez-vous décrire quelques exemples pertinents de son application concrète ?
– La complexité ou le coût de son utilisation sont-ils bien compensés par la qualité supplémentaire d'information qu'il permet d'obtenir ?

Ces questions sont très souvent posées par le jury, sous une forme ou une autre. Le jury invite les candidats à se préparer tout particulièrement à gérer ce type de questions, centrales dans la pédagogie de l'informatique au niveau des lycées et des classes préparatoires. On trouvera dans le rapport 2006 une discussion précise de quelques leçons à titre d'exemple.

7.2.2 Développement d'un point du plan

Lors de la présentation de son plan, le candidat propose au moins deux développements au jury. Parmi ceux-ci, le jury en choisit un.

Il n'est donc pas nécessaire de préparer des développements de très haut niveau. Au contraire, les candidats doivent s'entraîner à exposer des points simples du programme de manière pédagogique et stimulante. C'est particulièrement le cas s'il existe des livres usuels consacrés au thème de la leçon : les plans sont alors excellents... mais les développements souvent bien décevants !

Un point important dans la présentation du développement est de bien situer le problème considéré, de définir les notations utilisées et de préciser l'objectif visé, comme on le ferait pour un cours à des élèves. Attention à ne pas présenter la solution avant d'avoir bien expliqué le problème !

7.2.3 Interaction avec le jury

Environ la moitié de l'épreuve est consacrée à l'interaction avec le jury. En informatique, cette interaction ne prend pas la forme d'exercice d'application. Il s'agit plutôt d'explorer de manière plus approfondie les notions qui ont été présentées, les domaines connexes, et surtout les exemples d'application de ces notions.

L'interaction est conduite sous la forme d'un *dialogue* avec le candidat. Le jury respecte le niveau choisi par le candidat : les questions s'ajustent à ce niveau.

Ce long temps d'interaction doit être considéré comme une *chance* pour le candidat de montrer ses connaissances ! À lui de guider le jury dans la direction adéquate. Il est indispensable que les candidats s'entraînent à ce type d'exercice avec leurs préparateurs.

7.3 Remarques générales sur l'épreuve de modélisation analyse de systèmes informatiques

Le jury a apprécié le travail accompli pour la préparation de cette épreuve.

7.3.1 Présentation du texte

Les candidats ont en général compris la différence entre l'épreuve de modélisation et la leçon. Le candidat doit montrer qu'il comprend la modélisation décrite par le texte. Il n'est pas attendu que le candidat présente l'ensemble du texte, mais par contre il est attendu qu'il reste fidèle à l'esprit du texte et des développements proposés.

Le candidat dispose de 40 minutes pour décrire le problème, développer sa modélisation et présenter l'exercice d'informatique. Le jury n'est pas sensé connaître le texte à l'avance (même s'il en a une copie devant lui). C'est au candidat d'introduire le sujet du texte et de motiver la présentation qui va suivre. Cette motivation sera le plus souvent l'évocation de situations concrètes dans lesquelles on a besoin d'outils informatiques spécifiques. Ces situations peuvent être proposées par le texte lui-même, mais elle peuvent aussi être tirées de l'expérience personnelle du candidat. Toute contribution personnelle à ce niveau sera très appréciée !

Beaucoup de candidats omettent cette phase indispensable d'introduction et de motivation. Une manière simple de commencer est de présenter le plan de l'exposé. Ainsi, le jury pourra mieux se repérer par la suite, et éventuellement aider le candidat à mieux organiser son temps.

7.3.2 Exercice de programmation.

Au cours de l'exposé, le candidat présente son exercice de programmation. Pour cette partie de l'épreuve nous donnons quelques recommandations à la fin de ce rapport.

Cette partie de l'épreuve a été globalement satisfaisante, les candidats ayant généralement bien compris l'importance qui y est attachée. Elle dure environ 10 minutes. Le candidat choisit librement, dans les 40 premières minutes, le moment de présenter son exercice d'informatique, de façon qu'il s'intègre au mieux dans son exposé. Si l'exercice n'a pas été présenté au bout d'une trentaine de minutes, le jury lui rappellera de le faire.

Le plus souvent, les candidats le placent dès que les notions nécessaires ont été introduites dans l'exposé. Cette introduction doit être soignée et complète, afin d'éviter tant les allers-retours du terminal au tableau que les discours *avec les mains* devant l'écran.

Cette présentation au jury doit être faite que le programme *tourne* – ce que l'on espère – ou pas. Ensuite, le candidat lance une exécution. La possibilité de modification au vol d'un paramètre est appréciée pour la vérification de la correction. Dans tous les cas, que le programme *tourne* ou pas, le jury évalue la qualité

générale du code réalisé. Cette évaluation interactive permet à un candidat réactif de repérer une erreur, voire de la corriger, de recompiler et de relancer l'exécution.

7.3.3 Pédagogie

Le déroulement de l'épreuve de modélisation est très variable du fait des textes eux-mêmes. Les points d'attention du jury sont la présentation du plan, l'organisation du tableau, la lisibilité des écritures, la cohérence des notations, l'introduction et la motivation des définitions.

Dans cette épreuve, le candidat expose le résultat de sa réflexion en proposant un développement personnel. Un élément d'appréciation est le niveau des développements et la qualité des preuves qu'il choisit d'établir et de présenter.

7.3.4 Interaction avec le jury

Les règles générales sont les mêmes que pour l'épreuve de leçon d'informatique ci-dessus. Essentiellement, c'est le candidat qui détermine le niveau et la forme de l'interrogation, et il est important qu'il se saisisse de ce rôle.

Il est attendu du candidat qu'il prenne le temps de réfléchir aux questions ou aux suggestions qui lui sont proposées. Il est normal qu'un candidat réponde au jury qu'il ne voit pas comment répondre à une question et le jury fera bien sûr tout son possible pour lui permettre de reprendre pied dans le dialogue. Si en plus le candidat est capable de préciser *pourquoi* il ne voit pas ce qui est attendu de lui, cela simplifiera la tâche du jury.

Il est bien sûr attendu du candidat qu'il connaisse les notions utilisées dans le texte, mais aussi qu'il sache les utiliser ! En particulier, le jury ne fait pas de cloisonnement strict entre informatique et mathématiques. La réponse à des questions concernant les programmes, par exemple l'estimation du temps de calcul, peut nécessiter l'utilisation d'outils mathématiques du programme, par exemple des résultats simples de combinatoire ou d'analyse. Le jury demandera alors souvent de justifier tel ou tel choix de conception.

Le jury rappelle qu'il n'attend pas, dans les développements, un exposé, même brillant, d'une démonstration classique connue... et relue durant le temps de préparation !

7.4 Exercice de programmation informatique

Voici quelques recommandations plus précises concernant l'exercice de programmation. Elles sont motivées par les présentations des candidats de cette année. Nous espérons qu'elles seront utiles pour les candidats des années à venir.

D'une manière générale, le candidat doit proposer un code lisible et mettre en valeur ses connaissances en programmation. À titre de repère, la partie centrale du code devrait tenir sur un écran.

Les candidats sont invités à présenter le schéma algorithmique et les structures de données utilisés avant de lancer leur programme. Par contre, il est inutile de descendre dans les détails les plus triviaux du code, que le jury peut lire lui-même sur les écrans de contrôle. Le jury pourra demander au candidat d'évaluer la complexité de son implémentation ou de discuter de choix alternatifs de conception.

Le candidat choisit son langage. Cette année, nous avons vu peu de programmes en Java. Les candidats se partagent équitablement entre Caml et C (en fait, ils utilisent dans le langage C les extensions C++ admises par le compilateur `gcc`). Ce choix peut orienter les questions, car l'implémentation d'un problème peut être plus facile dans certains langages. Par exemple, la différence ensembliste entre deux listes étant prédéfinie dans les bibliothèques de Caml, on attend du candidat qui l'utiliserait qu'il puisse expliquer l'implémentation de cette fonction et la complexité des opérations concernées.

L'exercice est soigneusement spécifié dans les textes proposés. Il doit être conduit dans l'un des langages proposés (C, Caml ou Java) : un candidat qui n'utilise pas les langages proposés reçoit la note 0 à l'exercice de programmation.

De même, le candidat doit respecter la spécification qui est donnée dans l'énoncé. Il peut, s'il le souhaite, présenter ensuite un deuxième programme implémentant une autre spécification. Il devra alors expliquer pourquoi il le fait. Le fait que l'exercice proposé dans l'énoncé soit trivial ou inintéressant n'est évidemment pas une explication suffisante ! Ces extensions sont alors considérées et évaluées comme des développements au choix du candidat. Par exemple, des simulations simples ont pu servir à exposer un développement. Elles doivent mettre en valeur d'autres capacités du candidat que ses capacités en programmation.

Dans le cas d'une programmation en C, il sera systématiquement demandé au candidat de recompiler son programme avec le niveau maximal d'avertissement :

```
gcc -Wall prog.c -o prog
```

Un programme qui produit des avertissements sera pénalisé et le candidat devra le corriger pendant l'interrogation. La même chose sera vérifiée en Caml ou Java.

Certains candidats ont passé beaucoup de temps à programmer des entrées *interactives* au clavier. Ce n'est pas nécessaire et souvent inutilement complexe, notamment en C (appel par référence dans la fonction `scanf`, etc.). Il est recommandé de coder le jeu de données dans une procédure d'initialisation qui pourra être facilement modifiée à la demande du jury.

Il est demandé aux candidats d'exécuter leurs programmes sur différents jeux de données, et il est souhaitable qu'ils aient anticipé ce point. La manière dont ces jeux de données sont choisis devra être justifiée par la démonstration de divers aspects du comportement du programme. Les candidats sont souvent interrogés sur leurs critères de choix.

Il est très souvent demandé aux candidats d'exécuter leurs programmes sur les cas limites de leurs spécifications, sauf si ces cas ont été explicitement exclus dans la présentation préalable : liste vide pour les algorithmes de tri, nombres négatifs pour des algorithmes de factorisation, etc.

La lisibilité et l'élégance de l'expression sont particulièrement appréciées par le jury. Il est essentiellement attendu que le style de programmation des programmes soit *cohérent* : utilisation de structures d'itération (bornées `for` ou non-bornées `while`), initialisation des variables, découpage plus ou moins fin en fonctions auxiliaires, etc. Les critères d'arrêt des boucles doivent être parfaitement maîtrisés. Toutes les quantités présentes dans les programmes doivent être définies par des constantes symboliques.