Travaux pratiques d'informatique

Présentation de l'épreuve

L'épreuve de travaux pratiques d'informatique en MPI est une épreuve d'algorithmique et de programmation, avec utilisation d'un ordinateur, d'une durée de 3 heures. À partir d'un sujet imposé, elle demande de traiter sur machine des questions de programmation, d'effectuer des choix de modélisation et d'aborder certains aspects théoriques de l'informatique, tout en communiquant très fréquemment avec le jury.

L'objectif de l'épreuve est d'évaluer les capacités de programmation, la maitrise des méthodes classiques du programme, les capacités de modélisation, d'abstraction et d'inventivité ainsi que l'application de bonnes pratiques que l'on est en droit d'attendre de futurs ingénieures et ingénieurs.

Organisation de l'oral

Les candidats disposent d'un ordinateur fourni par le concours, configuré avec un environnement de travail adapté aux sujets demandés. Pour la session 2023, l'environnement choisi est la distribution GNU/Linux Ubuntu 22.04, construite de façon analogue à l'environnement Pronaos (https ://gitlab.com/agreg-info/clef-agreg/).

Avant le début de l'épreuve, le jury a laissé plusieurs minutes aux candidats afin de prendre en main librement cet environnement de travail. Pendant ce temps, qui n'entre pas en compte dans l'évaluation, il a été possible de poser toutes les questions nécessaires aux membres du jury. Le jury envisage de reconduire ce temps de 10 minutes de prise en main de l'environnement lors des sessions à venir.

Le jury a distribué un sujet imprimé aux candidats. Les sujets étaient tous accompagnés de fichiers auxiliaires, comprenant notamment :

— des fichiers sources, complets ou à compléter selon les cas ;

— des fichiers de données à exploiter ;

— des scripts d'aide à la compilation.

Ces fichiers sont mis à la disposition des candidats par le jury dans les premières minutes de l'épreuve, après la phase de prise en main et après la distribution des sujets papiers. Pour cette session, ils ont été mis à disposition sur une clé USB qui permettait également de sauvegarder le travail en cours tout au long de l'épreuve.

Il était demandé aux candidats de recopier dans leur dossier personnel les fichiers fournis sur les clés, de travailler dans le dossier personnel, puis, en fin d'épreuve de transférer l'ensemble des fichiers produits sur la clé USB.

Les consignes étaient clairement rappelées à chaque début de session et inscrites de manière bien visible et permanente dans les tableaux des salles.

Les sujets sont organisés en une suite de questions, à traiter généralement de façon linéaire. Ces questions sont réparties en trois catégories :

— des questions de programmation ;

— des questions à préparer pour une présentation orale ;

— des questions de rédaction à réaliser sur un compte rendu.

Le concours fournit toujours les feuilles et les brouillons utiles à l'épreuve. Les candidats doivent se munir du matériel d'écriture usuel (stylos, crayons, gomme, règle). Aucun autre matériel n'est autorisé. Il n'y a en particulier pas lieu de prévoir une calculatrice, les candidats disposant déjà d'un ordinateur.

Les sujets proposés suivent tous le format de l'épreuve 0 mise en ligne par le concours avant la session 2023.

Programme des épreuves orales

Les sujets posés permettent d'évaluer l'ensemble des notions présentes dans les programmes d'informatique des classes de MP2I et de MPI. Chaque sujet porte sur une ou plusieurs parties spécifiques de ces programmes. L'ensemble de tous les sujets couvrait globalement l'intégralité du programme des deux années.

Les langages évalués sont C, OCaml et SQL. Les sujets peuvent utiliser, selon la pertinence par rapport au thème abordé, un seul, deux ou trois langages (mais jamais SQL seul). La majorité des sujets impose le langage à utiliser pour chaque question, mais certaines parties peuvent être laissées librement au choix des candidats.

Évaluation des épreuves orales

Lors de cette session, le jury a proposé des sujets relativement longs, ce qui permettait à tous les candidats de se mettre en valeur et de démontrer leurs aptitudes. Les candidats les plus efficaces pouvaient traiter intégralement un grand nombre de parties, d'autres candidats étaient guidés par le jury vers une autre partie, par exemple en cas de blocage ou pour s'assurer d'évaluer toutes les compétences nécessaires.

L'évaluation de l'épreuve ne consiste cependant pas exclusivement, et loin de là, en une mesure du plus grand nombre de questions traitées. Le jury a souhaité valoriser non seulement l'efficacité en terme de programmation mais également l'intérêt porté à la discipline de programmation (compiler fréquemment, s'assurer que le code fonctionne, proposer spontanément des tests). Le jury considère qu'une épreuve pratique doit mener à un programme informatique qui fonctionne effectivement. Le jury a préféré des prestations avec un peu moins de questions traitées mais dans lesquelles les programmes écrits avaient été correctement compilés et soigneusement testés.

Les critères suivants ont reçu une attention particulière pour l'évaluation :

— la clarté des explications apportées au jury, le recul sur les notions abordées et sur les programmes proposés ;

— l'autonomie des candidats, au travers des compétences d'analyse des situations, de conception des solutions et de la mise en œuvre dans le langage demandé ;

— la compétence de modélisation, en utilisant de façon pertinente les connaissances et méthodes du programme ;

— la discipline de programmation, avec la clarté du code produit, la pertinence des structures choisies et la justification, notamment par des tests, de la correction des solutions choisies.

L'évaluation se fait sur les éléments présentés directement durant l'épreuve ainsi que sur les réponses portées sur le compte rendu. Le jury n'attend aucun formalisme particulier sur ce compte rendu et demande uniquement aux candidats d'écrire les réponses qu'ils souhaitent communiquer au jury. Les programmes écrits par les candidats sont relus directement par le jury pendant l'épreuve lors des échanges.

Le jury s'efforce d'être bienveillant en toute circonstance. Il fournit des indications et de l'aide, notamment pour éviter de laisser les candidats dans une situation de blocage. En revanche, le jury ne débogue pas

lui-même un programme qui ne fonctionne pas. Le jury peut cependant orienter le candidat vers les outils appropriés pour trouver l'origine de l'erreur.

Le jury interroge dans les limites du programme d'informatique de MP2I et MPI. Si certaines parties des sujets peuvent se placer ponctuellement à la limite de ce programme, il ne s'agit jamais de l'essentiel de l'évaluation, et ceci est réalisé en toute connaissance de cause. Par exemple, une question demandait de citer et d'expliquer le principe de deux algorithmes de tris, alors que seul le tri par tas — qui n'a d'ailleurs été que très peu cité — figure explicitement au programme. Aucun tri en particulier n'était donc attendu et tous étaient acceptés. L'évaluation portait principalement sur le recul, sur la qualité des explications et sur une analyse simple, raisonnable et cohérente de la complexité annoncée.

De manière générale, les sujets posés ont permis de bien différencier les prestations des candidats.

Analyse globale des résultats

Le jury a constaté avec plaisir que les prestations ont été dans l'ensemble de très grande qualité, démontrant d'excellentes capacités algorithmiques, un bon recul sur les notions du programme et de relativement bonnes capacités de programmation, de la part de pratiquement tous les candidats de la filière. Aucune prestation n'a été jugée comme étant réellement faible en niveau absolu. L'échelle de notation étant celle d'un concours, les plus basses notes rendent compte simplement d'une prestation moins bien classée par rapport aux autres.

Les candidats étaient bien préparés à l'épreuve. Personne ne semble avoir été surpris par le format de l'épreuve, que le jury a voulu conforme au sujet 0 publié en amont des épreuves orales. Ceci a permis lors des épreuves d'avoir des échanges entre le jury et les candidats de très haute qualité.

Le jury a constaté que dans la très grande majorité des cas, les compétences visées par les programmes de MP2I et MPI étaient acquises, que les aspects théoriques étaient maitrisés et que la technique était manipulée avec efficacité. Les candidats ont parfaitement démontré leurs aptitudes de futurs ingénieures et ingénieurs en informatique. Le jury les en félicite.

Commentaires sur les réponses apportées et conseils aux futurs candidats

Publication des sujets

Afin de renforcer la préparation à cette épreuve, le jury publie un échantillon de sujets qui ont été donnés lors de cette session accompagnés de remarques sur les attentes du jury.

Maitrise du cours

Le jury considère que la maitrise du cours est un élément essentiel : il encourage les candidats à bien *apprendre le cours* afin de traiter rapidement et efficacement les questions qui s'y rapportent. Les questions de cours ont souvent été discriminantes, les réponses parfaitement maitrisées contrastant fortement avec des réponses constituées d'éléments farfelus (à titre d'exemple, à plusieurs reprises, le principe de l'algorithme de Dijkstra n'a pas été compris, ou bien sa complexité n'a pas été évaluée correctement).

Le jury a constaté plusieurs confusions, les candidats ne connaissant pas les noms des algorithmes au programme. Le jury invite à faire preuve de précision, dans une optique d'efficacité de la communication. La simple confusion de noms n'a cependant pas été sanctionnée quand il était clair que le candidat maitrisait l'algorithme demandé et savait le décrire précisément. À contrario, redonner simplement en tant que mot-clé le nom d'un algorithme sans en connaitre le principe n'a pas été valorisé (surtout quand la

réponse est hors sujet, certains candidats ayant par exemple proposé Boyer et Moore en tant qu'algorithme de compression ou l'algorithme de Rabin-Karp pour illustrer la programmation dynamique).

Préparation technique

L'utilisation de l'environnement informatique mis à disposition n'a pas été une source de blocage majeur. Le jury a fourni, avant l'épreuve et pendant l'épreuve, les indications nécessaires aux candidats pour faire fonctionner cet environnement, en particulier sur des points non exigibles des programmes de MP2I et MPI (aide à l'invocation des compilateurs, rappels sur le système de modules en OCaml lorsque le sujet fournissait plusieurs modules, utilisation des scripts de compilation fournis, aide à l'accès aux outils de détection d'erreurs tels que l'*address sanitizer*, activation des avertissements du compilateur, etc.).

Le jury ne souhaite pas pénaliser les candidats en fonction des moyens informatiques qui ont été mis à leur disposition pendant leur formation, mais il constate cependant, malgré l'aide fournie pendant l'épreuve, une différence d'aisance marquée en faveur des candidats qui ont eu accès à un environnement adéquat. C'est pourquoi *le jury encourage vivement toutes les formations à proposer à leurs élèves, durant leur scolarité, un environnement de travail adapté au programme de MP2I et MPI*, permettant au minimum d'aborder les points suivants :

— l'accès à un *shell* dans un terminal permettant d'invoquer les compilateurs mais également d'autres commandes, et de manière générale permettant de se familiariser avec le fonctionnement d'une ligne de commandes ;

— le fonctionnement d'un système POSIX, en particulier quant à la gestion des processus, des fils d'exécution, des flux standard et leur redirection, éventuellement avec des tubes ;

— la familiarisation avec des outils de compilation usuels (`make` ou `dune`). Leur usage fera toujours l'objet d'un rappel et aucune compétence spécifique n'est attendue, mais avoir déjà rencontré ces outils permet une plus grande efficacité.

Le jury reprend à son compte l'indication suivante des programmes de MP2I et MPI : *« Bien que ces notions soient indépendantes du système d'exploitation, le système Linux est le plus propice pour introduire les éléments de ce programme. »*

Remarques concernant OCaml

Le jury maitrise la bibliothèque standard du langage mais il n'exige pas que les candidats utilisent d'eux-mêmes des fonctions de haut niveau, par exemple celles provenant du module `List`. L'évaluation est par exemple indifférente à l'utilisation, qui n'est ni valorisée ni sanctionnée, de `List.fold_left` (et autres fonctions similaires). Le jury est surtout attentif à la clarté du code produit, à sa correction et à la maitrise de ce code par les candidats. Certains codes ont été inutilement compliqués quand des solutions élémentaires existaient, ce qui a été la source de nombreuses erreurs d'exécution.

Certains sujets invitaient à utiliser dans un même programme des aspects fonctionnels et des aspects impératifs du langage, ce qui a déstabilisé plusieurs candidats ou bien a conduit à la rédaction de codes très compliqués. Le jury rappelle qu'un `match` est une expression comme une autre dans le langage et qu'il est parfaitement admis d'utiliser cette construction à l'intérieur d'une boucle `for`. Il est cependant impératif de bien connaitre les priorités des constructions et des opérateurs et de savoir parenthéser correctement (avec les mots-clés `begin-end` ou avec des parenthèses). Le jury conseille de systématiquement introduire ce parenthésage en présence conjointe de la construction `if then else` et de l'opérateur point-virgule (`;`), les erreurs de priorité à ce niveau ont posé d'innombrables problèmes à de très nombreux candidats, engendrant souvent une importante perte de temps.

De manière générale, la programmation impérative en OCaml est relativement mal maitrisée, avec des solutions bien plus compliquées que nécessaire, alors que le même programme en C ne poserait pas de problème.

Enfin, les candidats doivent savoir proposer un programme OCaml complet, qui compile dans son intégralité et pas uniquement ligne par ligne dans un REPL. Le jury accepte si nécessaire le double point-virgule (`;;`) pour séparer les phrases, même si ce dernier ne fait pas partie du langage à proprement parler. Les candidats doivent savoir identifier les erreurs de syntaxe dans leurs programmes, celles-ci se situent souvent bien avant la ligne à laquelle est indiquée l'erreur.

Remarques concernant C

La gestion de la mémoire est un aspect maitrisé par de nombreux candidats, mais les oublis d'allocations avec `malloc` et surtout les oublis de libération avec `free` ont été trop fréquents. Lors de la préparation des candidats, il peut être utile d'indiquer comment compiler un programme avec `gcc -g -Wall -fsanitize=address` et d'interpréter la sortie en cas d'erreur de segmentation ou de libération oubliée.

La compilation séparée a fait l'objet de rappels, mais le jury souligne que celle-ci doit être *reconnue* par les candidats qui sont donc supposés être un minimum familiers avec ce procédé.

Remarques concernant SQL

Un nombre non négligeable de candidats semble avoir fait l'impasse complète sur le langage SQL en sautant systématiquement les parties liées aux bases de données. Ceci n'indispose pas directement le jury mais prive, de fait, ces candidats de nombreux points, dont certains auraient été relativement faciles à obtenir.

Réalisation de tests

Les programmes demandés lors de l'épreuve de travaux *pratiques* sont destinés à être compilés et exécutés. Certains sujets demandent explicitement de fournir des tests dans des situations données (par exemple pour détecter une erreur dans un code fourni). Cependant, toute fonction écrite devrait faire l'objet de quelques tests (au minimum : une compilation et une exécution, et si c'est pertinent, l'ajout de quelques tests automatisés).

Les candidats ont adopté plusieurs approches de qualités différentes pour la réalisation de leurs tests :

— souvent en OCaml, par la saisie directe dans le REPL OCaml ou `utop`, d'appels de fonction ad-hoc et par comparaison manuelle des résultats ;

— par l'écriture d'une petite fonction principale réalisant des appels à `printf` sur le résultat de la fonction testée, et comparaison visuelle à l'exécution ;

— ou par l'utilisation de `assert` (ou similaire) dans une fonction de tests dédiée, qui compare automatiquement le résultat avec celui attendu. Cette approche est beaucoup plus robuste que les précédentes car elle est reproductible automatiquement, elle peut être relancée plus tard pour détecter des régressions et elle est source de moins d'erreurs.

Le jury a souvent interrogé les candidats sur la pertinence des tests, notamment en terme de couverture de code.

Temps d'appropriation de l'environnement

Les candidats sont invités à profiter pleinement du temps préalable d'appropriation de l'environnement, qui, rappelons-le, ne fait pas partie de l'évaluation. Il est en particulier utile de vérifier que l'on sait où

se trouve la documentation des trois langages, de s'assurer que l'on arrive à compiler correctement un programme simple et d'ouvrir son éditeur préféré pour vérifier que tout fonctionne correctement. Signalons qu'il n'est pas du tout interdit, pendant ce temps de prise en main, de commencer à programmer de petites fonctions utilitaires, qui peuvent toujours servir plus tard, ce qui permet également de commencer à prendre ses marques.

Gestion du temps et de la progression

La plupart des sujets sont conçus pour être traités de manière linéaire en suivant l'ordre des questions, ce que le jury impose en général, sauf mention explicite du contraire. Il est toujours possible de passer certaines questions, le jury se réservant le droit d'approuver cette initiative ou de demander de revenir à une question non traitée. Le jury propose parfois, et impose dans certains cas, de changer de partie ou de sauter une ou plusieurs questions, cela, toujours dans l'intérêt du candidat et de son évaluation.

Interaction avec le jury

La qualité de l'échange avec le jury est un élément déterminant de l'évaluation. Le jury passe régulièrement auprès de chaque candidat en s'assurant d'un traitement équitable. Il peut être utile de signaler au jury que l'on dispose d'éléments à porter à sa connaissance, mais il n'est pas utile de manifester son impatience si celui-ci n'est pas immédiatement disponible.

Les candidats sont encouragés à utiliser certaines de leurs feuilles de brouillon comme support lors de leurs interactions avec le jury. Il ne s'agit pas de rédiger les questions à développer à l'oral, mais d'utiliser à bon escient un support écrit, par exemple pour proposer des formules, des schémas ou des dessins.

Conclusion

Le jury a été globalement été très satisfait par l'ensemble des prestations des candidats, qui sont généralement très bien préparés aux modalités de cette épreuve alors même qu'il s'agissait de la première édition suite à la création d'une nouvelle filière, avec un programme ambitieux et conséquent. Le jury tient en particulier à féliciter l'ensemble des candidats ainsi que leurs enseignants pour leur engagement, leur implication et l'aboutissement de ces années de travail.

Quitte à se répéter, le jury insiste une nouvelle fois sur l'importance de la compréhension et de la maitrise du cours, sur la nécessaire prise de recul quant aux notions abordées pendant l'année et sur l'intérêt d'une pratique très régulière sur machine tout au long du cursus, afin de pouvoir aborder le plus sereinement possible l'épreuve proposée.