

Le langage Caml aux concours

BATOG Guillaume

30 août 2006

Écrit Polytechnique

Concours 2002

12 questions dont 7,5 questions de programmation Caml (63%)

La concision d'un programme est un point important. Cependant ne pas confondre concision avec lisibilité. La sanction est d'autant plus grande si le code est abandonné au correcteur sans explication de l'algorithme sous-jacent. Inversement, il est inutile de justifier plus que nécessaire une fonction dont l'algorithme est évident d'après l'énoncé (premières questions). L'excès de justifications qui se révèle inexacte la plupart du temps, n'est pas sanctionné. C'est une mauvaise gestion du temps. On demande donc au candidat d'ajouter ses explications là où il les pense utiles pour le correcteur. Les erreurs de syntaxe sur un code bien indenté ne sont que très faiblement sanctionnées. Dans certains cas cependant (fonction dont le code est trivial), elles peuvent être sanctionnées plus lourdement et faire la différence entre deux copies équivalentes. Voici quelques erreurs et codes inutiles relevés : « ; else », « | p when p=0 -> ... », « !i=!i+1 », T_(i+1).

Concours 2003

17 questions dont 9 questions de programmation Caml (53%)

La plupart de ceux qui programment en Caml réinventent à longueur de copie `map`, `do_list` et `it_list`. Peut-être n'utilisent-ils pas les fonctions de bibliothèque de peur de se tromper dans leur maniement, ou de susciter l'ire du correcteur ; une façon très simple de dissiper ces craintes (systématiquement employée dans les meilleures copies) consiste à donner en annexe une implémentation de ces fonctions.

Dans cette épreuve, les erreurs de syntaxe et de type les plus fréquentes étaient des erreurs vénielles relatives aux fonctions de bibliothèque `vect_item` et `vect_assign`, dont beaucoup de candidats n'avaient probablement pas entendu parler (aucune copie ne mentionne même le nom de ces fonctions).

D'autre part, l'identification de motifs est trop systématiquement employée, donnant lieu tantôt à des complications ridicules comme « `match (a,b) with (x,y) when x=y -> ...` » pouvant conduire à l'erreur classique « `match (a,b) with (a,a) -> ...` », tantôt à des erreurs révélant une incompréhension de la notion (j'ai ainsi relevé un curieux « `match n with p+1 -> p` »).

Pour les questions comportant des programmes à écrire, les critères principaux étaient la correction de l'algorithme sous-jacent et l'effectivité du code (tout programme non conforme aux spécificités de l'énoncé, aux erreurs de syntaxe et aux étourderies près, a été noté en dessous de la moyenne) ; les critères secondaires étaient l'efficacité du programme (en particulier les fonctions d'étiquetage de complexité quadratique ou pire ne pouvait rapporter que la moitié des points de la question), sa lisibilité et sa justification (quand le passage des spécifications à l'algorithme n'était pas évident).

Autres concours

Concours 2004 : 17 questions dont 11 questions de programmation Caml (65%)

Concours 2005 : 18 questions dont 11 questions de programmation Caml (61%)

Concours 2006 : 15 questions dont 12 questions de programmation Caml (80%)

Épreuve pratique aux ENS

3h30 sur machine (50%) – 20 min de présentation orale (50% de la note)

L'objectif de cette épreuve est d'évaluer les capacités des candidats à mettre en oeuvre la chaîne *complète* de résolution d'un problème informatique : analyse des spécifications abstraites, conception d'algorithme et choix des structures de données, évaluation de sa complexité (coût en temps et en mémoire), programmation sur machine dans l'un des langages proposés, tests des programmes sur de petites valeurs des paramètres, et exécution sur des valeurs précises des paramètres. À la fois algorithmique, pratique et orale, cette épreuve est donc transversale et permet de tester la maîtrise du candidat sur la résolution concrète d'un problème informatique et de sa capacité à en présenter la solution. (...) Quelques remarques d'ordre général :

- Les questions sont calibrées pour qu'elles nécessitent entre un et dix millions d'itérations pour les plus grandes valeurs des paramètres, soit moins d'une seconde d'exécution avec l'algorithme attendu.
- Préférer la structure de tableau à toute autre structure de données lorsque le nombre d'entrées est connu, et n'utiliser les listes chaînées ou les arbres que lorsque les tableaux sont inadaptés. En effet, la structure de données de tableaux est bien plus facile à maintenir et à contrôler, donc à débbugger.

La plupart des questions proposaient plusieurs tailles pour le problème à résoudre. Elles admettaient quasiment toujours un algorithme naïf qui permettait de résoudre les tailles moyennes (par exemple en $O(n^4)$). En revanche seuls les algorithmes efficaces (par exemple en $O(n^2)$) permettaient de résoudre les tailles les plus grandes. Les tailles les plus petites étaient généralement solubles à la main et étaient posées pour permettre (inciter) aux candidats de tester leur programme.

Nous constatons que les candidats sont mieux préparés d'une année sur l'autre, ce qui est le signe que l'épreuve est entrée dans les moeurs. En particulier, la plupart des résultats résolubles à la main étaient corrects. Nous rappelons la nécessité d'apprendre aux candidats à tester et surtout à débbugger leurs programmes (par exemple en ajoutant des commandes d'affichage, ne serait-ce que pour vérifier que l'exécution du programme avance). Contrairement aux années passées, la largeur des sujets a été largement revue à la baisse pour être maintenant traitables dans une large partie par les meilleurs des candidats dans le temps imparti, les examinateurs considérant que les candidats disposent maintenant d'un stock suffisant de sujets pour leur préparation. Pour plus d'informations, le site web de l'épreuve : <http://www.ens-lyon.fr/LIP/ConcoursInfo/>

Autres écrits

Le sujet 2005 de Centrale était constitué de deux problèmes indépendants. Le premier comportait une grande part de programmation. Il a permis aux candidats sérieux de tirer leur épingle du jeu. (...) La partie programmation donne toujours lieu aux maladroites habituelles marquant le manque de pratique (manipulation maladroite de booléens, matchings maladroits ou incorrects, manipulation des listes comme des vecteurs...) La manipulation des types enregistrement a été assez négligée. Ces types sont peut-être moins utilisés (dans les énoncés de concours) que les fonctions, listes et autres vecteurs, mais sont pourtant de première importance. (...) Signalons enfin aux candidats que faire du filtrage sur un booléen peut-être avantageusement remplacé par un `if ... then ... else...`. Le sujet 2004 ne comportait aucune question de programmation.

Le sujet des Mines est généralement constitué de deux problèmes dont l'un porte sur l'algorithme. Pas de remarques particulières concernant la programmation dans les rapports du concours.

TIPE... ?