I Remarques générales

- Il peut faire très chaud dans les salles d'oraux : pensez à prendre de l'eau et, éventuellement, un brumisateur.
- Toutes les ressources (rapports, annales...) sont sur le site du cours.

TP

- Vous pouvez tester les environnements de TP des concours à l'aide d'une machine virtuelle.
- Penser à sauvegarder régulièrement.
- Savoir utiliser le terminal : `cd` pour se placer dans le bon dossier, `ls` pour afficher les fichiers du répertoire courant, Ctrl + C pour arrêter un processus (boucle infinie)...
- Savoir compiler et interpréter du code : `gcc fichiers.c`, `ocamlc fichiers.ml`, `utop`, Makefile (voir indications CCINP)...
- Savoir exécuter du code : `./a.out` pour exécuter l'exécutable produit par `gcc` ou `ocamlc`.

Messages d'erreurs classiques :

- `undefined reference` : une fonction utilisée n'est pas définie (potentiellement `main`). Vous avez peut-être oublié de mettre tous les fichiers .c nécessaires en argument à `gcc`.
- `syntax error` : erreur de syntaxe (parenthèses, points-virgules...).
- `segmentation fault` : accès à une zone de mémoire non autorisée (dépassement de tableau, pointeur non initialisé...).

Avec `-fsanitize=address`, le message d'erreur est plus précis et donne la ligne (`main.c:12` signifie ligne 12 du fichier `main.c`) :

- `heap-buffer-overflow` : dépassement de tableau alloué sur le tas (avec `malloc`).
- `stack-buffer-overflow` : dépassement de tableau alloué sur la pile.
- `use-after-free` : accès à une zone de mémoire après l'avoir libérée avec `free`.
- `double-free` : tentative de libérer une zone de mémoire déjà libérée.
- `memory-leak` : une zone de mémoire allouée n'est jamais libérée (il manque un `free`).
- `stack-overflow` : dépassement de la pile, généralement dû à une récursion infinie.

II CCINP

Format

- Exercice type A (/8) : théorique. SQL possible.
- Exercice type B (/12) : programmation en OCaml/C sur ordinateur (pas SQL).

Déroulement

- 15 minutes de prise de connaissance de l'exercice A.
- 30 minutes de préparation du sujet sur feuille (exercice A) et ordinateur (exercice B). Il faut être suffisamment rapide pour avancer dans l'exercice B pendant ce temps de préparation.
- 27 minutes de passage au tableau. Le code pour l'exercice B est projeté au tableau. Il peut être demandé de continuer à programmer pendant le temps de passage. CCINP met à disposition un Makefile qui est sur le site du concours.

Conseils du jury

- Sont prises en compte dans l'évaluation : maîtrise du programme, de l'environnement technique et des outils mis à disposition, pertinence de la réflexion, interactivité lors de la présentation des résultats et des tests (exercice de type B), réactivité aux éventuels conseils et indications de l'examinateur, qualité de la prestation orale.
- De nombreux candidats n'ont pas su gérer leur temps durant le passage, passant par exemple plus de 20 minutes sur l'exercice A et réfléchissant à des questions qu'ils n'ont pas abordées en préparation plutôt que de présenter ce qu'ils ont fait sur l'exercice B d'abord. Un candidat qui réussit brillamment l'exercice A et n'aborde pas l'exercice B ne pourra avoir une note supérieure à 8/20.

- Il est recommandé aux candidats de commencer par annoncer, en début d'épreuve, les questions des deux exercices qui ont été traitées en préparation. Ceci permet à l'examinateur d'aider les candidats à gérer leur temps et de les interroger sur l'ensemble des questions abordées pendant la phase de préparation.
- Le candidat peut, bien entendu, modifier son code pendant l'oral pour corriger ses erreurs ou aborder des questions non traitées en préparation en les programmant devant l'examinateur sur la machine de la salle.
- Il est attendu des candidats une maîtrise élémentaire d'un terminal de manière à pouvoir se déplacer dans une arborescence et lancer une commande de compilation de leur code. Sans être nécessaire, l'examinateur rappelle que, dans un terminal, la touche de tabulation permet souvent de compléter des morceaux de la commande en cours et que les flèches directionnelles permettent de naviguer dans l'historique des commandes.
- Trop nombreux sont les candidats qui arrivent au passage sans avoir jamais évalué leur code en préparation. Cela se voit dès la première question de code. Ils passent alors un temps laborieux à corriger la question 1 pendant le passage et se retrouvent parfois à ne même pas pouvoir présenter le code des questions suivantes.
- Les tests des fonctions de l'exercice B font partie de l'évaluation et l'examinateur demande à voir le résultat de ces tests pendant le passage. Il est souhaitable d'écrire les tests à l'avance plutôt que de devoir les réécrire pendant le passage.

III Mines-Télécom

Format

Deux exercices, sans programmation OCaml/C (mais peut contenir du SQL ou pseudo-code) :

- Exercice 1 : proche du cours, contenant souvent un algorithme ou structure de données à décrire et appliquer.
- Exercice 2 : plus long et théorique.

Déroulement

- 15 minutes de préparation sans note.
- 30 minutes de passage. Les exercices sont à traiter dans l'ordre.

Conseils du jury

- Lors de l'épreuve orale sont évaluées, non seulement les connaissances en informatique, mais aussi le dynamisme et la capacité à interagir avec le jury, à l'écouter et à rebondir sur ses remarques et indications. Le jury déconseille fortement aux candidats de se murer dans le silence trop longtemps face à une question difficile. Tant qu'il ne fait pas d'erreur de cours ou de raisonnement grossière, le candidat ne peut que gagner à partager ses idées et ses pistes de résolution avec le jury lorsqu'il entre dans une phase de réflexion.
- Le jury fournit une liste non exhaustive de questions de cours qui sont apparues cette année dans les exercices pour faciliter le travail de révisions des futurs candidats (voir rapport sur le site du cours).

IV Mines-Pont

Format

Environ 4 candidats composent en parallèle sur ordinateur, un examinateur circule parmi eux.
Les candidats doivent fournir un compte rendu servant à signaler leur progression à l'examinateur (indiquer quelles fonctions permettent de répondre à telle question, expliquer brièvement le fonctionnement de celles qui sont délicates, indiquer les tests effectués...). On y indique aussi les questions posées à l'oral par l'examinateur et les réponses apportées.
Les interactions avec l'examinateur, le code produit (tests compris) et le compte-rendu sont évalués.

Déroulement

- Court temps de familiarisation avec l'environnement.
- TP de 3h30 sur ordinateur.

Conseils du jury

- Au cours des 3h30, les examinateurs passent régulièrement voir les candidats. Les candidats qui ont une question peuvent appeler les examinateurs sans attendre que ceux-ci ne passent. Certains candidats tentent de sauter certaines parties sans le demander à l'examinateur. Cette technique est à proscrire lors d'un oral, elle exhibe de toute façon les lacunes du candidat (peur de la technicité, impasse, ou autre).
- Même quand les candidats ont la bonne idée, ils ont du mal à décrire et donc à réfléchir sur leur solution avant de se lancer dans le code. Cela pénalise souvent les candidats qui perdent du temps à ne pas tester chaque partie indépendamment quand il y a des bugs, qui écrivent souvent des bouts de code inutiles qu'ils doivent ensuite retravailler et parfois qui se rendent compte qu'au bout de 15 minutes que leur solution ne marche pas. Pour toutes les questions non triviales, le jury conseille de passer quelques minutes à élaborer rapidement la solution (par exemple sous forme de pseudo code ou juste en donnant les grandes étapes) puis à réfléchir à comment simplifier la solution ou la décomposer en plusieurs sous-problèmes.
- Pour les questions d'algorithmique non triviales, il était attendu des candidats qu'ils testent leurs programmes.
- Il est presque toujours recommandé d'utiliser les warnings du compilateur et les options de compilation recommandées en C sont `-Wall`, `-Wextra` et `-fsanitize=address`.
- Nous encourageons vivement les candidats à tester la machine virtuelle proposée sur le site du concours, ne serait-ce que 10 min, pour ne pas être déstabilisé face à l'environnement qu'ils rencontreront le jour du concours.
- L'utilisation de ocamlopt ou de ocamlc peut se révéler laborieuse et le manque d'aisance fait perdre du temps aux candidats. À l'inverse, certains candidats ne savent utiliser OCaml qu'en compilant. Dans certains sujets, utiliser un shell (comme utop ou même ocaml utilisé en mode interactif) est un grand atout. Rappellons l'existence de `#use "fichier.ml"` qui permet d'évaluer intégralement un fichier.
- Quand le nom de la fonction et de ses arguments ne suffisent pas à comprendre son objectif, ou si la fonction est longue avec plusieurs blocs qui accomplissent plusieurs choses différentes, le jury attend aussi un commentaire rapide qui décrit ce que fait le bout de code considéré.
- Il est important de bien maîtriser et de savoir implémenter rapidement tous les algorithmes de base (parcours en largeur et en profondeur, tas...), mais il faut aussi connaître toutes les définitions et algorithmes du programme.
- Certains candidats ne savent pas rattraper une exception en OCaml (soit parce qu'ils ne connaissent pas `try` et `with` soit parce qu'ils ne savent pas qu'il faut que les types du blocs try et with soient compatibles). La documentation sur les langages est fournie et les candidats qui ont oublié certains points de syntaxe peuvent souvent s'en sortir seuls.

V CentraleSupélec

Format

- Trois types de questions : questions de programmation à résoudre sur machine, questions à préparer pour une présentation orale, questions à traiter sur papier.
- Ce n'est pas (que) une épreuve de programmation : les connaissances théoriques sont aussi évaluées.

Déroulement

- 15 minutes de familiarisation avec l'environnement.
- TP de 3h30 sur ordinateur.

Conseils du jury

- Avant le début de l'épreuve, le jury laisse une dizaine de minutes aux candidats afin de prendre en mains librement cet environnement de travail, de bien choisir son éditeur, de vérifier qu'il sait interpréter ou compiler en OCaml et en C et de se familiariser avec la documentation qui est proposée avec Zeal pour les langages OCaml/C/SQL. Pendant ce temps, qui n'entre pas en compte dans l'évaluation, il est possible de poser des questions techniques sur l'environnement aux membres du jury, de s'échauffer en écrivant de petits programmes simples ou encore de préparer des fonctions utilitaires générales.
- Les candidats sont responsables des sauvegardes de leur travail, qu'ils doivent effectuer à intervalles réguliers.
- Le compte rendu sert uniquement de support pendant l'interaction orale et est évalué par le jury directement pendant l'épreuve. On n'attend pas une rédaction détaillée comme on pourrait en trouver sur une copie à l'écrit, d'éventuelles imprécisions pouvant être levées par la discussion orale.

- Trop de candidats ne sont pas capables de restituer avec exactitude un algorithme ou un théorème de cours, ni de l'utiliser correctement. Les preuves des théorèmes sont à connaître. Le lemme de l'étoile est très rarement restitué ou utilisé correctement. Les réductions sont très rarement faites correctement, alors même qu'elles sont guidées ou très simples. La connaissance du cours prenant une part importante dans la note, le jury encourage vivement les candidats à retravailler les algorithmes au programme et les démonstrations classiques pour cet oral.

VI ENS

Format

- Oral théorique.
- TP algorithmique pour option informatique (ou Saclay et Rennes en option mathématiques).

Oral théorique

- 30 minutes de préparation.
- 28 minutes de passage au tableau.

TP

- 10 minutes de prise en main de l'environnement de TP.
- 3h30 de TP sur machine.
- 20 minutes de présentation devant un jury et rendu d'une fiche réponse.

Les TP sont différents des autres concours : s'entraîner sur les exemples corrigés du site officiel et lire le rapport.

Conseils du jury

- Il est souhaitable de décrire (oralement ou/et par un dessin) une preuve dans les grandes lignes avant de se lancer dans la preuve formelle. Cela permet à l'examinateur de s'assurer que le candidat a compris le principe de la preuve. De même, il est souhaitable de décrire un algorithme dans les grandes lignes avant de se lancer dans l'écriture du pseudo-code.
- Certains sujets demandent d'absorber plusieurs notions nouvelles, parfois au moyen d'un formalisme assez lourd. On attend alors des candidats une capacité à s'affranchir du formalisme pour se concentrer sur la signification des objets. Les candidats doivent d'abord les assimiler, et se former rapidement une intuition.
- (TP) Nous invitons les candidats à se familiariser à l'avance avec la manière dont ces suites pseudo-aléatoires sont utilisées dans les sujets précédents afin de gagner du temps le jour de l'épreuve.

VII École Polytechnique

Oral théorique

48min de passage au tableau, sans préparation.

Conseils du jury

- Il convient à la fois d'être suffisamment précis pour convaincre l'examinateur que l'on a compris ce qui se passait, mais de ne pas passer trop de temps sur la question sauf si l'examinateur le demande. Par exemple, il ne faut pas hésiter à donner des éléments de preuve directement à l'oral plutôt que de systématiquement les écrire : si l'examinateur constate que le candidat a manifestement compris ce qu'il avance et que la solution proposée répond à la question, il pourra ainsi inviter le candidat à passer immédiatement à la question suivante.
- Afin de mieux manipuler les objets manipulés pendant l'oral, il est souvent très pertinent de faire des dessins et de regarder de petits cas. Ainsi, trop de candidats ont commencé à s'embourber en utilisant des notations parfois absconses, jusqu'au moment où l'examinateur leur a demandé de dessiner le graphe ou la fonction qu'ils étudiaient, ce qui les a très souvent débloqués. De même, face à une question fermée, plusieurs candidats sont partis bille en tête sur une conjecture formulée, semble-t-il, au hasard, au lieu de commencer par manipuler de petits exemples.
- Il est évidemment nécessaire de connaître parfaitement son cours. Cette année, par exemple, plusieurs candidats se sont fourvoyés sur la complexité de l'algorithme du parcours en profondeur.