Quelques conseils pour les TP d'informatique

1 Conseil généraux, déroulé

- Prévoir du papier pour prendre des notes au cours de la lecture du sujet, et éventuellement griffoner lors des phases de recherche de solution.
- Bien lire l'énoncé (conseil toujours valable !). Le rapport des épreuves CCINP mentionne notamment l'importance de bien lire le sujet afin de respecter les types imposés par le sujet et pour éviter de re-coder des fonctions qui sont en fait données dans le fichier compagnon.
- Si le langage est laissé au choix, décider quel langage utiliser après avoir parcouru tout le sujet, pour avoir une bonne idée des enjeux.
- Ouvrir dès le début à la fois un éditeur de texte et le terminal. La séance doit être succession de passage de l'un à l'autre.
- Utiliser un éditeur de texte connu, qui permet, a minima, la coloration syntaxique et la numérotation des lignes. On profitera utilement des options d'indentation automatique, surlignage des parenthèses correspondantes et auto-complétion.
- En C, commencer par inclure les librairies utiles et définir la fonction `main`.
- Après chaque définition de fonction ou de type : compiler ou interpréter avant de passer à la définition suivante.
- Pour compiler en C, utiliser a minima l'option `-Wall` pour les avertissements et `-o <outputfile>` pour fixer le nom de l'exécutable. Les options `-Wextra` et `-fsanitize=adress`, qui seront rappelées le jour du TP, permettent davantage d'avertissements. Lorsqu'un `Makefile` est fourni, ne pas s'en priver.
- En cas de multiples erreurs, commencer par résoudre la première erreur, et recompiler alors.
- Dès que possible, coder un jeu de tests pour chaque nouvelle fonction (le plus souvent juste après la définition de la fonction, avant de se lancer dans autre chose). Penser à lancer ce jeu de tests (exécution en C, et si besoin appel de la fonction en OCaml). Le rapport du CCINP est tout à fait explicite : "le test des fonctions demandées par l'exercice B fait partie de l'évaluation".
- Pour déboguer :
  - penser à mettre un `'\n '` à la fin des affichages, sans quoi l'absence d'affichages temporisés peut induire en erreur ;
  - ne pas hésiter à indiquer ce qu'on affiche (`"i=\%d\n "` plutôt qu'un simple `"\%d\n "`), cela simplifie grandement l'interprétation des affichages par la suite.
- Sauvegarder régulièrement (ce qui en principe est déjà fait si on compile souvent).
- En cas d'usage d'une clé USB, travailler avec les fichiers sur l'ordinateur pendant la préparation, à la fin de l'épreuve enregistrer tous les fichiers sur clé, et éjecter la clé avant de la débrancher de la machine.
- En cas de présentation orale après la préparation, garder à portée de main la clé USB, sa convocation et sa pièce d'identité lors du changement de salle afin de ne pas perdre de temps.

2 Maintenir son code propre et correct (objectifs convergents)

- Se conformer aux noms de fonction et de type fixés par l'énoncé.

- Utiliser des noms de variables et de fonction évocateurs.
- Commenter son code :
    · typer toutes ses fonctions, en C et en OCaml ;
    · décrire de manière concise et précise ses fonctions (résultat et éventuels effets de bords) ;
    · le cas échéant, indiquer les hypothèses sur les arguments de la fonction ;
    · dès que possible indiquer des invariants, qui éclairent le rôle des variables ;
    · préciser le rôle des variables pour lesquelles il n'est pas évident ;
    · indiquer dès la définition de type ou de `struct` les hypothèses que doivent vérifier les champs pour former un objet valide ;
    · indiquer lorsqu'une fonction C alloue un espace mémoire qu'il faudra libérer plus tard.
- Veiller à ne former que des objets valides, à ne pas casser cette validité par effet de bord.
- Veiller à libérer l'espace mémoire alloué.

3 Gagner en agilité

- Utiliser sans réserve les raccourcis clavier (plus rapides que la souris) :
    · copier : Ctrl+C dans la plupart des éditeurs, Ctrl+Maj+C dans `utop` et le terminal ;
    · coller : Ctrl+V dans la plupart des éditeurs, Ctrl+Maj+V dans `utop` et le terminal ;
    · couper : Ctrl+X dans la plupart des éditeurs ;
    · rechercher : Ctrl+F dans la plupart des éditeurs ;
    · rechercher-remplacer : Ctrl+H avec `Gedit`, comprendre l'utilité des options mots entiers, sensible à la casse ;
    · enregistrer : Ctrl+S dans la plupart des éditeurs ;
    · enregistrer sous : Ctrl+Maj+S dans la plupart des éditeurs ;
    · commenter la sélection : dépend des éditeurs, peut s'avérer très pratique.
- On peut aussi utilement utiliser :
    · le double clic pour sélectionner un mot, plutôt que cliquer/glisser/lâcher ;
    · la touche Maj + les flèches pour sélectionner une zone contigüe ;
    · les touches Tab ou Maj+Tab pour indenter ou rétro-indenter un bloc de lignes sélectionnées.
- Utiliser la flèche du haut dans `utop` et dans le terminal pour remonter l'historique des commandes.
- Utiliser l'autocomplétion dans `utop` et dans le terminal.