Liste des projets

1. Interprète LISP — (seul ou en binôme)

Écrire en C un programme capable d'exécuter du Lisp standard. Les programmes Lisp peuvent être :

- directement écrits en ligne de commande ;

  Le code est interprété une fois que toutes les parenthèses ouvertes ont été fermées. Ceci permettrait d'écrire des fonctions qui tiennent sur plusieurs lignes sans passer par un éditeur de texte.

- chargés et exécutés à partir d'un fichier.

L'interprète doit :

- avoir une gestion dynamique de la mémoire ;
- gérer les fonctions arithmétiques ;
- inclure `t, nil, undefined, car, cdr, cons, quote, print, set, if, cond, eq, atom, while` et le plus important, **defun** (ou **de**) ce qui permettrait de créer les fonctions manquantes directement en Lisp et les charger au démarrage de l'interprète (ex. : `append, length, caar, cadr,` ...) ;

2. Calculatrice — (seul ou en binôme)

Écrire en C une calculatrice scientifique en ligne de commande. Vous pouvez prendre pour exemple **bc** (saisir : `$ bc -l` dans un terminal). Cette calculatrice doit, d'une part, gérer des fonctions telles que : `sqrt, pow, cos, sin,` ... (voir les fonctions présentent dans `math.h`), puis d'autre part, manipuler des variables ainsi que quelques instructions conditionnelles (`if, while,` ...).

3. Rechercher - Remplacer — (seul)

Nous souhaitons développer un outil (une commande `shell`) facilitant les opérations de rechercher/remplacer dans des fichiers texte. Cet outil prend plusieurs entrées possibles :

- -f fichier1 [fichier2 [...]] : pour rechercher dans les fichiers cités ;
- -d répertoire1 [répertoire2 [...]] : pour rechercher dans les fichiers présents dans les répertoires cités ;
- -R : (uniquement avec -d) pour effectuer une recherche récursive dans les répertoires cités ;
- mask="un masque" : (uniquement avec -d) pour restreindre la recherche seulement aux noms de fichiers correspondant à "masque". Exemple : mask="*.c" demande de rechercher dans les fichiers `C` ;
- find="mot1" : rechercher le pattern "mot1" ;
- replace="mot2" : remplacer par le pattern "mot2".

Seule restriction : ne pas utiliser d'appel aux commandes Shell.

4. Agenda — (seul)

Nous souhaitons avoir un programme de gestion d'agendas. Nous pouvons Ajouter/Modifier/Supprimer des contacts. Chaque agenda est lu/enregistré à partir d'un fichier texte. Les contacts sont représentés sous la forme de listes chaînées. Nous avons pour chaque contact, les champs suivants : nom, prénom, adresse (num. de rue, rue, code postal, ville, pays), tél. dom., tél. bureau, fax, email — aucun doublon n'est permis. Tous ces champs peuvent servir à effectuer des recherches. Pour cela un pré-tri sur chaque champs doit être effectué (exemple :*Quick Sort* puis recherche dichotomique). Pour préserver la mémoire, chaque liste triée suivant un champs ne doit contenir que les pointeurs vers les contacts.

5. Puissance 4 — (seul)

Créez un programme qui PERMET DE JOUER et JOUE au jeu Puissance4. On suppose que la largeur de la grille ainsi que le nombre de pièces devant être alignées pour gagner la partie sont définis au début de chaque partie par le joueur. Remarque : la hauteur de la grille est illimitée.

6. Calculs matriciels — (seul)

Écrire en C un programme qui pour des matrices quelconques, effectue des additions, soustractions, multiplications et inversions matricielles.

7. Jeu de rôle — (seul ou en binôme)

On souhaite écrire un moteur de jeu de rôle. Dans ce jeu, plusieurs personnages peuvent évoluer dans un environnement totalement décrit dans un fichier externe. L'interface du jeu est en mode texte. À chaque déplacement, l'interface décrit la scène en cours (ex. : un arbre en face, un chemin à l'est, une grotte sombre à gauche ...). Le personnage peut effectuer certaines actions : Aller, Prendre, Utiliser ... Biensûr, les actions impossibles ne doivent pas être permises (ex. : Prendre la grotte ou Aller à l'est s'il n'y a pas de chemin à l'est).

8. Métro — (seul ou en binôme)

Créez un programme qui à partir d'une station de métro de départ, trouve le chemin permettant de se rendre à une station de métro d'arrivée. Ce programme doit utiliser les listes chaînées. On prendra comme exemple une partie du métro parisien (Lignes 1, 2, 3, 4, 5 et 6). La définition des lignes de métro est donnée dans un fichier texte ; chaque ligne est définie par un temps moyen entre chaque station (ex. 90s) et dans l'ordre, les noms des stations. Les intersections des lignes (les correspondances) seront calculées automatiquement ; chaque correspondance a un coût de 5mn. On pourra demander soit le chemin le moins coûteux en temps soit le chemin comportant le moins de correspondances.

9. Nombres premiers — (seul)

Écrire un programme qui calcule (et affiche) à l'infini la suite des nombres premiers. Utilisez, pour la gestion des grands nombres, la librairie **GMP**. La librairie et sa documentation se trouvent à l'adresse : "http ://www.swox.com/gmp/".

10. π — (seul)

Écrire un programme qui calcule (et affiche) à l'infini les décimales du nombre π. Utilisez, pour la gestion des grands nombres, la librairie **GMP**. La librairie et sa documentation se trouvent à l'adresse : "http ://www.swox.com/gmp/".

11. Le nombre e — (seul)

Écrire un programme qui calcule (et affiche) à l'infini les décimales du nombre e ($2,7182...$). Utilisez, pour la gestion des grands nombres, la librairie **GMP**. La librairie et sa documentation se trouvent à l'adresse : "http ://www.swox.com/gmp/".

12. Le plus court chemin dans un labyrinthe — (seul)

Écrire un programme qui génère aléatoirement des labyrinthes connexes (il existe au minimum un chemin reliant deux points) de taille quelconque. Trouver (vous pouvez utiliser l'algorithme de Dijkstra) le plus court chemin reliant deux points du labyrinthe.

13. Gestion du personnel d'un restaurant — (seul ou en binôme)

On doit gérer au mieux le bon fonctionnement d'un Fast-Food. Le restaurant comporte au maximum : 6 postes de cuisinier, 12 postes de caissier, 3 postes à l'entretient et 1 poste de *manager* (Cette configuration est donnée comme exemple et pourrait être modifiée). Aux heures de pointes (Le *Rush*), 12h00-14h00 et 19h00-21h00, le restaurant doit fonctionner à plein régime (remplir le maximum de postes possibles.). Le reste du temps, le nombre de personnes par type de poste peut être divisé par 2 ou 3 (sauf pour le *manager*). Le restaurant est ouvert de 10h00 à 0h00 et on est dans une politique de 35 heures / semaine (permission de 39 heures pour 20% du personnel chaque semaine). Votre personnel est composé par exemple de : 7 polyvalents (cuisine - caisse - entretient), 20 caissiers, 18 cuisiniers et 3 *managers*. Une journée de travail ne dépasse pas 9h ; l'heure de pause étant comprise. La semaine est de 5 jours mais le restaurant est ouvert 7j/7. On donnera un identifiant unique pour chaque membre de l'équipe. Écrire le programme qui se charge, chaque semaine, de calculer le planning.

14. Gestion d'une file d'attente dans le service des urgences — (seul ou en binôme)

Nous devons gérer le service des urgences d'un hôpital. Les patients se présentent aléatoirement avec des maux plus ou moins grâves. On fixera la gravité et le temps indispensable aux soins de chaque mal sur des échelles de 1 à 10. On fixera aussi le nombre de médecins ou infirmières nécessaires pour chaque type de maux. Les maux les plus grâves étant mortels s'ils ne sont pas traités immédiatement. Vous n'avez que N médecins et M infirmières. Chaque tour de boucle représente 15mn à l'échelle humaine. Vous pouvez avoir toutes les 15mn, un nombre de nouveaux patients variant aléatoirement, entre 0 et 3 patients. Gérez la file d'attente.

15. U.M.T.S. — (seul)

Vous avez la possibilité de charger une image satellite de dimension quelconque. Cette image est une carte des altitudes du terrain sur lequel une société de Télécoms souhaite placer N antennes de relais. Sachant que ces antennes ne couvrent qu'un rayon de largeur R et qu'elles n'émettent pas vers le haut (une demi sphère), écrire le programme qui calcule le meilleure emplacement pour ces N antennes de façon à obtenir la meilleure couverture possible.

16. Jeu de Shannon — (seul)

Soit un graphe, dont deux noeuds (D et A) sont particuliers et deux joueurs, le lieur et le casseur. À chaque tour de jeu, le casseur casse une arête entre deux noeuds. À chaque tour de jeu, le lieur rend incassable une arête (non cassée) entre deux noeuds. Si, à un moment du jeu, il n'y a plus aucun moyen d'aller de D à A, le casseur a gagné. Si, à un moment donné, il existe un chemin incassable reliant D à A, le lieur a gagné.
Faites un programme qui permet à deux joueurs de jouer.
Faites un programme qui permet de jouer contre la machine.

17. Arbre Généalogique — (seul)

Créer un programme qui gère un arbre généalogique. On pourra faire des recherches dans l'arbre généalogique, afficher les «`fils de, fille de, père de, mère de ...`» , ou bien effectuer des tris par noms ou par date de naissance par exemple. Les seules structures de données admises sont les listes chaînées.

18. Gestion d'un stock de denrées périssables — (seul ou en binôme)

On doit gérer, pour le compte de plusieurs hypermarchés, le stock d'un hangar contenant plusieurs types de produits périssables (sacs de pomme de terre, pack de lait ...). Ces produits sont regroupés par type et date de péremption. Des opérations de dépôt/retrait sont effectuées tous les jours. Pour chaque jour, nous avons à notre disposition un fichier contenant la liste des produits à retirer/déposer. Attention ! les produits à retirer ne sont pas forcément disponibles (l'erreur est humaine), il faut donc en tenir compte et le signaler dans un fichier de LOG. À tout moment, on pourra demander au programme de nous sortir la liste (quantité - date - type) des produits disponibles dans le hangar ; on pourra aussi faire des recherches par date et type.

19. Gestion de comptes bancaires — (seul)

Nous devons créer un logiciel de gestion de comptes banquaires. Chaque compte doit avoir un identifiant unique qui le lie à une personne (nom, prénom, adresse ...) ansi qu'à une banque (identifiant, nom, adresse). Le logiciel doit effectuer et enregistrer toutes les opérations standards liées à un compte chèque : dépôt espèces ou chèque, virement émis ou reçu, paiement CB ou chèque, retrait CB et enfin ouverture ou fermeture de compte. Pour accéder à un compte, nous pouvons soit entrer directement son numéro, soit effectuer une recherche par nom (voir nom-prénom) afin d'obtenir les numéros de comptes correspondants. Nous pouvons, à tout moment, demander un historique complet (dates et natures des opérations suivies du solde du compte) des opérations effectuées sur un compte. Évidemment, toute opération doit être enregistrée immédiatement dans un même fichier afin de pouvoir récupérer les données d'une exécution à une autre.

20. Mots croisés — (seul)

Partant d'un fichier "dictionnaire" contenant des lignes "mot : définition", nous souhaitons écrire un programme qui génère une grille de mots croisés de dimensions quelconques (l'utilisateur spécifiera les dimensions dans la ligne d'arguments du programme). À chaque execution, la grille générée doit être (si possible) différente des précédentes.

21. Format BMP, PNG et JPG — (seul)

Écrire une bibliothèque qui permet de lire et d'enregistrer des fichiers BMP, PNG et JPG. Pour tester cette bibliothèque, nous allons lire le fichier, dessiner sur l'image correspondante une diagonale et l'enregistrer dans un autre fichier.