Examen - ING1 GI
Programmation C

Modalités

- Durée : 3 heures
- Toutes vos affaires (sacs, vestes, trousse, etc.) doivent être placées à l'avant de la salle.
- Aucun document papier n'est autorisé.
- Aucune sortie n'est autorisée avant une durée incompressible de 1 heure.
- Aucun déplacement n'est autorisé.
- **Aucune question au professeur n'est autorisée**. Si le sujet ne vous semble pas clair, à vous d'expliquer dans des commentaires pourquoi est ce que ça ne vous semble pas clair, et quelles modifications vous effectuez pour réaliser l'exercice.
- Aucun échange, de quelle que nature que ce soit, n'est possible.
- Les différentes fonctions/procédures demandées seront toutes à tester dans un **main**.
- Vous traiterez chaque exercice dans un dossier différent.
- Le barème est indicatif, il intègre pour chaque question un test dans un main.
- La présence d'un Makefile et des commentaires doxygens sont un plus, et seront appréciés.
- La lisibilité du code et les commentaires seront pris en compte dans la notation, de même que la séparation des différentes procédures/fonctions dans des fichiers d'en-tête (.c .h).
- **Tout code qui ne compile pas ne sera pas corrigé et entraînera une pénalité de 50%. Si une partie de votre code ne compile pas, commentez-le. Ceci implique évidemment que vous compiliez au fur et à mesure!**
- Chaque **warning** de compilation entrainera une **pénalité de 15%**.
- Tout ce qui sera dans le dossier rendu correspondra à votre rendu, et sera récupéré à la fin de l'examen. Je vous conseille vivement de travailler directement dans ce dossier.

Exercice : Jeu du pendu (1.5 + 3 + 3 = 7.5)

Le *Pendu* est un jeu consistant à trouver un mot en devinant quelles sont les lettres qui le composent. Le joueur a le droit à 10 propositions mauvaises avant de se faire pendre (et de perdre).
Exemple d'exécution (partielle):

```
Mot : _________         Nb d'essai restant : 10
Quelle lettre voulez-vous jouer ?   A
Mot : _________         Nb d'essai restant : 9
Quelle lettre voulez-vous jouer ?   E
Mot : ____E___E         Nb d'essai restant : 9
Quelle lettre voulez-vous jouer ?   I
Mot : ____E___E         Nb d'essai restant : 8
Quelle lettre voulez-vous jouer ?   O
Mot : __O_E___E         Nb d'essai restant : 8
```

1. Écrire la fonction `int joueLettre (char* motATrouver, char* motMystere, char lettre)` qui prend en paramètre `motATrouver` le mot qui doit être deviné par le joueur, `motMystere` le mot qui est affiché à l'utilisateur et `lettre` la lettre que l'utilisateur veut jouer, qui vient faire apparaître dans `motMystere` la `lettre` donnée et qui renvoie le nombre de fois que la `lettre` est apparu dans le `motMystere`.
2. Écrire la procedure `void jouePendu (char* motATrouver)` qui prend en paramètre le mot à faire deviner et qui déroule le jeu. Les différentes étapes du jeu sont :
    a. Faire une copie du mot et remplacer tous les caractères de la copie par un caractère marquant la non-découverte de la lettre
    b. Tant que tout le mot n'a pas été découvert et qu'il reste encore des essais
        i. Demander une lettre
        ii. Jouer cette lettre
    c. Annoncer si le joueur a gagné (en combien de coups) ou s'il a perdu.
3. Afin de faire varier les mots à deviner, le mot à trouver sera tiré aléatoirement dans un fichier. Le fichier aura en première ligne le nombre total de mots qu'il contient, puis un mot par ligne.
   Écrire la fonction `char* motAleatoire (char* nomFichier)` qui renvoit un mot prit aléatoirement dans le fichier.

Remarque : Un fichier "mots.txt" est fourni dans le dossier `docs/` contenant une liste de mots prédéfinis.

Exercice : Combat de Magiciens (1 + 2 + 1 + 2 + 2,5 = 8,5)

Dans le monde de Havnui, une petite équipe de magiciens tente de vaincre les démons présent sur leur planète. Les démons présent ont beaucoup de points de vie (PV) mais aucun points de magie (PM). A l'opposé, les magiciens ont plus de PM que de PV. Les magiciens n'ont que deux types de sorts : soin ou attaque. Ils peuvent faire une attaque physique mais elle est beaucoup moins efficace qu'un sort d'attaque.

Nous allons implémenter un combat entre une équipe de 4 magiciens et un démon.
Le démon aura des PV compris entre 3500 et 4000. Les magiciens auront des PV compris entre 200 et 300 et des PM compris entre 500 et 700.
Le démon attaque aléatoirement un des joueurs de l'équipe (non décédé), et ses attaques infligent entre 60 et 80 points de dégâts.
Les sorts d'attaque des magiciens infligent entre 80 et 130 points de dégâts et leur coûtent 30 PM, et leurs attaques physique infligent entre 20 et 30 points de dégâts. Les sorts de soin coûtent 20 PM et restaurent 30 PV mais ils ne peuvent pas ressusciter quelqu'un.
Lors d'un combat, les magiciens n'ont pas l'information du nombre de PV du démon.

1. Définir une structure de magicien contenant son nom, ses PV et PM.
   Définir une structure de démon contenant son nom et ses PV.
2. Ecrire une fonction qui permet de créer un démon.
   Ecrire une fonction qui permet de créer une équipe de 4 magiciens.
3. Ecrire une procédure qui permet d'afficher un démon et l'équipe.
4. Ecrire la fonction de combat (version 1 puis version 2) entre l'équipe de magiciens et le démon. La méthode renvoie 1 si les magiciens ont été victorieux, 0 sinon.

Combat :
Le combat aura jusqu'à ce que les magiciens aient gagnés le combat ou qu'ils soient tous décimés. Les magiciens combattent à tour de rôle, et le démon attaque après le 2e et le 4e magicien.

a. *Version 1 : que de l'attaque !*

Dans cette première version, les magiciens ne feront que attaquer (donc pas de soins). S'ils ont suffisamment de PM, ils lanceront un sort. Sinon, ils font une attaque physique.

b. *Version 2 : attaque et soin !*

A leur tour de jeu, les magiciens pourront choisir entre un sort d'attaque sur le démon ou un sort de soin sur un des membres de l'équipe.

Exercice : Statistiques (1 + 1 + 2 = 4)

Nous souhaitons faire un peu de statistiques sur les notes des étudiants à l'EISTI. Les notes seront stockées dans un tableau, la taille et les valeurs seront saisies par l'utilisateur.

1. Écrire la fonction `saisirNotes` qui prend en paramètre la taille du tableau, qui demande les notes à l'utilisateur et qui renvoie le tableau de notes.
2. Écrire la fonction `moyenne` qui calcul et renvoie la moyenne des notes stockées dans le tableau.
3. Écrire la fonction `variance` qui calcul et renvoie la variance des notes stockées dans le tableau.

Rappel :

La formule de la variance σ^2 est donnée en fonction de la moyenne $\bar{x}$ par : $\sigma^2 = \frac{\sum(x-\bar{x})^2}{n}$