CESI
ÉCOLE D'INGÉNIEURS

Examen en langage C

Une fiche A4 préalablement validée est autorisée.
L'usage de l'intelligence artificielle et des poissons rouges savants sont strictement interdits.
Celui de l'intelligence humaine est vivement recommandé.

Vous disposez d'un solde de 3 jokers : 

EXERCICE 1 : AU CDI ***(8 points, 1 point par question)***

Utiliser le fichier cdi.c comme base pour l'exercice.

Pour créer son site en ligne, le CDI du collège-lycée fait appel à un élève ingénieur du CESI. Il a besoin d'afficher les manuels scolaires et leurs informations et de gérer son stock.

a. Créer une structure `Auteur` (nom, prénom).

b. Créer une structure `Manuel` (une matière, une classe, un auteur, une collection, un éditeur et une année de parution, un stock).

c. Créer un tableau de 4 manuels scolaires et les initialiser avec les informations du tableau suivant.

Matière	Classe	Auteur	Collection	Éditeur	Année	Stock
Hist. Géo	6ème	Stephan Arias	Magellan	Magnard	2019	32
Français	5ème	Hélène Potelet	Rives Bleues	Hatier	2010	1
Maths	3ème	Jules Malaval	Transmaths	Nathan	2021	28
Physique-Chimie	Terminale	Lucien Bernard	Sirius	Nathan	2020	12

d. Écrire une fonction qui prend en paramètre un manuel et qui affiche ses informations.

e. Écrire une fonction qui prend en paramètre un tableau de manuels et qui permet d'afficher leurs informations *(indice : faire appel à la fonction précédente).*

f. Écrire une fonction permettant de modifier le stock suite à la vente d'un manuel passé en paramètre. L'appliquer au manuel de Français de 5ème.

g. C'était le dernier manuel de Français pour les 5ème ! Le CDI a reçu un nouveau stock de 50 manuels réédités (2024). Mettre à jour manuellement le stock puis écrire une fonction permettant de modifier l'année de publication d'un manuel passé en paramètre. L'appliquer au manuel en question.

h. Écrire une fonction qui affiche les informations du livre le plus récent.

EXERCICE 2 : RÉALISATION D'UNE BATAILLE NAVALE ***(17 points)***

Utiliser le fichier batnav.c comme base pour l'exercice.

Le but de l'exercice est de programmer un jeu de la bataille navale (jeu où s'affrontera un humain contre l'ordinateur) en faisant **une analyse descendante**.

Les règles sont les suivantes :

- Chaque joueur dispose d'une grille (10 × 10),
- Les colonnes sont repérées par des lettres (A à J) et les lignes par des numéros (1 à 10),
- Chaque joueur place 5 navires, horizontalement ou verticalement :
  - un **porte-avions** (5 cases), symbolisé par l'entier 1
  - un **croiseur** (4 cases), symbolisé par l'entier 2
  - un **contre-torpilleurs** (3 cases), symbolisé par l'entier 3
  - un **sous-marin** (3 cases), symbolisé par l'entier 4
  - un **torpilleur** (2 cases), symbolisé par l'entier 5

Le but de chaque joueur est de couler tous les bateaux de l'autre joueur. Chaque joueur joue tour à tour en proposant une position où lancer un explosif pour toucher un navire adverse en indiquant une position sur la grille (J2 par exemple) et l'adversaire répond *Touché* si la torpille touche un bateau, *Coulé* si l'adversaire touche un bateau et le coule (c'est-à-dire qu'il a touché toutes les cases du bateau correspondant) ou *À l'eau* si l'explosif n'a rien touché ou a touché un emplacement d'un bateau déjà coulé.

La bataille navale à Lagos le 27 juin 1693, Théodore Gudin

Une grille sera encodée informatiquement par un tableau 2D de 10 × 10 entiers avec le codage suivant :

- 0 indique qu'il n'y a pas de bateau
- un entier entre 1 et 5 indique qu'il y a un bateau (1 pour porte-avions, 2 pour croiseurs, etc.)
- 6 indique la présence d'un endroit où un bateau a déjà été touché

On souhaite aussi avoir un système d'authentification bancal qui ne permet d'accéder au jeu que si l'utilisateur a saisi le bon mot de passe (un code à 4 chiffres : `2709`).

Toutes les fonctions devront être testées dans le programme principal.

a. Écrire une fonction `verifier_mdp` qui demande à l'utilisateur de rentrer le mot de passe tant qu'il est incorrect. Le programme doit être intuitif et afficher des messages à l'utilisateur. Dès qu'il est connecté, la fonction doit afficher *« Bienvenue à la Bataille Navale »*.

b. Créer une structure `Bateau` (nom, taille et numéro). Créer aussi les 5 bateaux du jeu et les ranger dans un tableaux bateaux de telle-sorte que `bateaux[1]` soit le bateau symbolisé par l'entier 1. `bateaux[0]` peut être vide.

c. Créer deux grilles `grille_1` et `grille_2`. Écrire une fonction `init_gille` qui prend en paramètre une grille et l'initialise à 0 (pas de bateaux placés).

d. Créer une fonction `affiche_grille()` qui prend en paramètre une grille puis réalise un bel affichage identique à celui de l'exemple ci-dessous. Une case avec bateau sera affichée avec le numéro du bateau, tandis qu'une case sans bateau (0) sera affichée avec une vague (le caractère « ~ »). Il faudra faire attention à l'alignement. Pour gérer l'alignement des nombres (10 étant à deux chiffres), on pourra utiliser `%2d` dans la fonction `printf`.

   Exemple d'affichage :

```
    A B C D E F G H I J
 1  ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
 2  ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
 3  ~ ~ ~ ~ 5 5 ~ ~ ~ ~
 4  ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
 5  ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
 6  ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
 7  ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
 8  ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
 9  ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
10  ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
```

e. En C, le premier index d'un tableau est 0, ce qui n'est pas très intuitif pour un humain. Écrire deux fonctions `convertir_ligne` et `convertir_colonne` qui prennent en paramètre respectivement un numéro de ligne (entre 1 et 10) et une lettre de colonne (entre 'A' et 'J') et qui retournent un numéro de ligne (ou de colonne) entre 0 et 9. Si la ligne et/ou la colonne n'appartiennent pas à la grille, les fonctions doivent retourner -1.

f. Écrire une fonction `positionne` qui prend en paramètre une grille, un numéro de ligne (entre 1 et 10), une lettre de colonne (entre 'A' et 'J'), un sens (une lettre 'h' ou 'v') et un bateau (de la classe `Bateau`). Dans le cas où il est possible de positionner le bateau, la fonction doit modifier la grille et renvoyer 1. Dans le cas inverse, s'il n'est pas possible de le positionner, elle doit renvoyer -1.

   Les numéros de ligne et de colonne correspondant au point le plus « en haut à gauche ». Il faut vérifier que le bateau ne sort pas de la grille et qu'il ne chevauche pas un autre bateau.

   Horizontalement (respectivement, verticalement), un bateau ne peut être placé si la somme de son numéro de colonne (ligne) et de sa taille est supérieure à la taille d'une ligne (colonne). Faire des schémas si nécessaire pour plus de compréhension.

g. Écrire une fonction `initGrilleSurDemande` qui prend en paramètre une grille et qui la fait remplir par l'utilisateur. Pour chaque bateau, on demande à l'utilisateur de saisir d'abord la lettre, puis le nombre, puis la direction ('h' pour horizontal, 'v' pour vertical), en répétant la question si l'utilisateur saisit une réponse incompréhensible et tant qu'il ne saisisse pas une bonne valeur (par exemple une lettre qui n'est pas comprise entre A et J, ou un mauvais nombre pour la ligne ou un mauvais caractère pour la direction).

h. Écrire une fonction `est_coule` qui prend en paramètre un bateau (de la classe `Bateau`) et une grille et qui renvoie 1 si le bateau est coulé (c'est à dire que son numéro n'apparaît plus dans la grille) et 0 sinon.

i. Écrire une fonction `attaque` qui prend en paramètre une grille, une colonne et une ligne et qui affiche les messages « Touché », « Coulé » ou « À l'eau » correspondant au résultat d'une attaque sur la position repérée par la ligne et la colonne. La fonction devra aussi le cas échéant mettre la grille à jour en indiquant quelle case a été touchée.

j. Écrire une fonction `initGrilleHasard` qui prend en paramètre une grille et qui la remplit aléatoirement.

   Rappel : En C, pour avoir un entier aléatoire entre min et max, il faut d'abord inclure les bibliothèques `time` (`#include <time.h>`) et `stdlib` (`#include <stdlib.h>`), puis initialiser dans le `main` la fonction de génération aléatoire (`srand((unsigned int)time(NULL));`). Il sera ensuite possible d'utiliser l'instruction suivante, qui retourne un entier : `rand() % (max - min + 1) + min`.

k. Écrire une fonction `a_perdu` qui prend en paramètre une grille et qui renvoie 1 si le joueur possédant la grille a perdu (plus aucun numéro de bateau) et 0 sinon.

l. Écrire deux fonctions `joue_ordi_idiot_ligne` et `joue_ordi_idiot_colonne` qui renvoient au hasard un numéro de ligne entre 1 et 10 et une lettre de colonne entre A et J.

m. Créer une structure `Joueur` (pseudo et score). Créer aussi les deux joueurs de la partie (l'utilisateur et l'ordinateur). Les initialiser avec un pseudo vide (" ") et un score nul.

n. Écrire une fonction `maj_score` qui met a jour le score d'un joueur mentionné en paramètre.

o. Écrire une fonction `choisir_pseudo` qui prend en paramètre un pseudo et un joueur et qui permet d'attribuer le pseudo au joueur.

p. Écrire une fonction `lancer_jeu_joueur_ordi` en faisant appel aux différentes fonctions écrites qui permet à un utilisateur d'affronter l'ordinateur. Une partie se termine dès qu'un joueur a perdu.

q. Améliorer « l'intelligence » de l'ordinateur en tenant compte des coups précédemment joués (ainsi que des résultats de chaque coup). L'ordinateur ne jouera pas plusieurs fois au même endroit et saura adapter sa stratégie, si notamment une case visée n'était pas vide. Écrire ainsi une ou deux fonctions `joue_ordi_malin`.

r. Écrire une fonction `lancer_jeu_joueur_joueur` qui permet de lancer une partie entre deux joueurs.

s. Écrire une fonction `lancer_jeu_ordi_ordi` qui permet de lancer une partie entre deux intelligences et qui permet à l'utilisateur de les observer jouer.

t. Écrire le code principal permettant à l'utilisateur de choisir son mode de jeu (contre un ordinateur ou contre un autre joueur), le cas échéant, le niveau du jeu (facile contre un ordinateur idiot, difficile contre un ordinateur malin) et le nombre de manche et qui lance le jeu.

Attention : la propreté du code et les explications (en commentaires) seront pris en compte lors de la notation.

Prénom : _____ Nom : _____ Pseudo : _____

Grille d'avancement et de notation

Complétez cette grille au fur et à mesure de votre avancement. Une question non marquée ne sera pas prise en compte lors de la notation.

Vous disposez d'un solde de 3 jokers : 

EXERCICE 1 : AU CDI

	Fait ?	Note
a. Créer une structure `Auteur` (nom, prénom).		
b. Créer une structure `Manuel` (une matière, une classe, un auteur, une collection, un éditeur et une année de parution, un stock).		
c. Créer un tableau de 4 manuels scolaires et les initialiser avec les informations du tableau suivant.		
d. Écrire une fonction qui prend en paramètre un manuel et qui affiche ses informations.		
e. Écrire une fonction qui prend en paramètre un tableau de manuels et qui permet d'afficher leurs informations.		
f. Écrire une fonction permettant de modifier le stock suite à la vente d'un manuel passé en paramètre. L'appliquer au manuel de Français de 5ème.		
g. C'était le dernier manuel de Français pour les 5ème ! Le CDI a reçu un nouveau stock de 50 manuels réédités (2024). Mettre à jour manuellement le stock puis écrire une fonction permettant de modifier l'année de publication d'un manuel passé en paramètre. L'appliquer au manuel en question.		
h. Écrire une fonction qui affiche les informations du livre le plus récent.		
Note totale de l'exercice 1		

EXERCICE 2 : RÉALISATION D'UNE BATAILLE NAVALE

	Fait ?	Note
a. Écrire une fonction `verifier_mdp` qui demande à l'utilisateur de rentrer le mot de passe tant qu'il est incorrect.		
b. Créer une structure `Bateau` (nom, taille et numéro). Créer aussi les 5 bateaux du jeu et les ranger dans un tableaux bateaux.		
c. Créer deux grilles `grille_1` et `grille_2`. Écrire une fonction `init_gille` qui prend en paramètre une grille et l'initialise à 0 (pas de bateaux placés).		
d. Créer une fonction `affiche_grille()` qui prend en paramètre une grille puis réalise un bel affichage identique à celui de l'exemple ci-dessous.		
e. Écrire deux fonctions `convertir_ligne` et `convertir_colonne` qui prennent en paramètre respectivement un numéro de ligne (entre 1 et 10) et une lettre de colonne (entre 'A' et 'J') et qui retournent un numéro de ligne (ou de colonne) entre 0 et 9.		

f. Écrire une fonction `positionne` qui prend en paramètre une grille, un numéro de ligne (entre 1 et 10), une lettre de colonne (entre 'A' et 'J'), un sens (une lettre 'h' ou 'v') et un bateau (de la classe `Bateau`). Dans le cas où il est possible de positionner le bateau, la fonction doit modifier la grille et renvoyer 1. Dans le cas inverse, s'il n'est pas possible de le positionner, elle doit renvoyer -1.		
g. Écrire une fonction `initGrilleSurDemande` qui prend en paramètre une grille et qui la fait remplir par l'utilisateur.		
h. Écrire une fonction `est_coule` qui prend en paramètre un bateau (de la classe `Bateau`) et une grille et qui renvoie 1 si le bateau est coulé (c'est à dire que son numéro n'apparaît plus dans la grille) et 0 sinon.		
i. Écrire une fonction `attaque` qui prend en paramètre une grille, une colonne et une ligne et qui affiche les messages « Touché », « Coulé » ou « À l'eau » correspondant au résultat d'une attaque sur la position repérée par la ligne et la colonne.		
j. Écrire une fonction `initGrilleHasard` qui prend en paramètre une grille et qui la remplit aléatoirement.		
k. Écrire une fonction `a_perdu` qui prend en paramètre une grille et qui renvoie 1 si le joueur possédant la grille a perdu et 0 sinon.		
l. Écrire deux fonctions `joue_ordi_idiot_ligne` et `joue_ordi_idiot_colonne` qui renvoient au hasard un numéro de ligne et une lettre de colonne.		
m. Créer une structure `Joueur` (pseudo et score). Créer aussi les deux joueurs de la partie (l'utilisateur et l'ordinateur).		
n. Écrire une fonction `maj_score` qui met a jour le score d'un joueur mentionné en paramètre.		
o. Écrire une fonction `choisir_pseudo` qui prend en paramètre un pseudo et un joueur et qui permet d'attribuer le pseudo au joueur.		
p. Écrire une fonction `lancer_jeu_joueur_ordi` en faisant appel aux différentes fonctions écrites qui permet à un utilisateur d'affronter l'ordinateur. Une partie se termine dès qu'un joueur a perdu.		
q. Écrire une ou deux fonctions `joue_ordi_malin`.		
r. Écrire une fonction `lancer_jeu_joueur_joueur` qui permet de lancer une partie entre deux joueurs.		
s. Écrire une fonction `lancer_jeu_ordi_ordi` qui permet de lancer une partie entre deux intelligences.		
t. Écrire le code principal du jeu.		
Note totale de l'exercice 2		

Note totale de l'examen		/25
		/20
Observation :		