Centre de Tele-enseignement Universitaire
UNIVERSITÉ DE FRANCHE-COMTÉ

Vous devez rendre votre devoir sous forme d'une archive zip contenant les documents demandés lors des exercices.

Exercice 1 : Knowledge is power (5 points)

Question 1 : R épondez aux questions ci-dessous :

Les réponses sont à rendre dans un fichier texte, au nom de l'exercice, dans un des formats suivants : .txt, .odt, .doc(x), .pdf

1. En C, à quoi sert le mot clé static ?
2. En C++, qu'est-ce que le Name Mangling ?
3. Quel est le problème avec l'instruction suivante : *if (a=5)* ?
4. Qu'est-ce qu'une classe abstraite ?
5. Qu'est-ce qu'un pointeur ?
6. Dans une classe, à quoi sert le mot clé private ?
7. Qu'est ce que la déclaration suivante : *char *v[8];*
8. Si je crée un tableau d'entiers comme suit : *int tab[3] = 1, ;*. Que va contenir la variable *tab* ?
9. Citez un framework de tests unitaires.
10. En programmation, qu'est-ce qu'une librairie statique ?

Exercice 2 : Analyse (5 points)

Voici quelques lignes de code, écrites en C++.

```
1  #include <iostream>
2  using namespace std;
3
4  static void f(char *c, int start, int count)
5  {
6      char *ptr = c + (start * sizeof(char));
7      while (NULL != ptr && count - - > 0)
8      {
9          (*ptr) = 'x ';
10         ptr += sizeof(char);
11     }
12 }
13 static int g(int **c)
14 {
15     if (NULL == c)
16     {
17         return -1;
18     }
```

```
19      if (NULL == *c)
20      {
21          return -2;
22      }
23      if (0 == (**c % 2))
24      {
25          return **c;
26      }
27      return -3;
28  }
29  int main(void)
30  {
31      char a[] = {'0 ', '1 ', '2 ', '3 ', '4 ', '5 ', '6 ', '7 ', '8 ', '9 '};
32      int b = 5;
33      int *p = NULL;
34      f(a, 2, 4);
35      cout << a << endl;
36      f(a, 4, 5);
37      cout << a << endl;
38      f(a, 0, 1);
39      cout << a << endl;
40      cout << g(&p) << endl;
41      p = &b;
42      cout << g(&p) << endl;
43      b = 2;
44      cout << g(&p) << endl;
45      *p = 10;
46      cout << g(&p) << endl;
47      return 0;
48  }
```

Les réponses sont à rendre dans un fichier texte, au nom de l'exercice, dans un des formats suivants : .txt, .odt, .doc(x), .pdf

Question 1 : Que font les méthodes f et g ?

Question 2 : Que fait la ligne : char *ptr = c + (start * sizeof(char)) ; ?

Question 3 : Que va-t-il s'afficher, exactement, à l'exécution ?

Exercice 3 : Bataille navale (10 points)

Le dossier *battleship* contient les fichiers sources d'un jeu de bataille navale. Pour rappel, le jeu de bataille navale [1], est un jeu de société dans lequel deux joueurs doivent placer des «navires» sur une grille tenue secrète et tenter de «toucher» les navires adverses. Le gagnant est celui qui parvient à couler tous les navires de l'adversaire avant que tous les siens ne le soient. On dit qu'un navire est coulé si chacune de ses cases a été touchées par un coup de l'adversaire.

Le programme est découpé en plusieurs fichiers, sources et en-têtes, tous présents dans le répertoire *sources*. Le point d'entrée est le fichier *main.cpp*, il ne contient que quelques instructions pour initialiser le moteur de jeu, et démarrer le jeu.

1. source : `https://fr.wikipedia.org/wiki/Bataille_navale_%28jeu%29`

Le moteur de jeu est dans le fichier *game.cpp*, et l'en-tête associé *game.h*. Le moteur de jeu contient la préparation du jeu, puis la logique de jeu (*BattleShipGame : :play*).

Il existe deux types de joueurs, humain et ordinateur, le code propre à chaque joueur est dans les fichiers *human.cpp* et *computer.cpp*. La classe donnée dans *player.cpp* sert d'interface pour représenter un joueur. Un joueur possède un plateau de jeu, *BattleShipBoard gameBoard* et cinq navires, *Ship *ships[5]*. Les navires sont créés à la construction d'un joueur, par contre ils ne sont pas placés dans la grille de jeu du joueur. C'est la fonction *start* qui doit s'en charger. Pour un joueur humain, il faut inviter le joueur à entrer des coordonnées (A0, E4, etc.), un sens (horizontal, vertical), puis placer le navire dans la grille. Cette opération est répétée pour tous les navires. Pour un ordinateur, le placement est à votre convenance.

La fonction *play* permet aux joueurs de joueur. C'est le moteur de jeu qui va appeler cette fonction en alternant le joueur un, puis le joueur deux.

Pour un humain la fonction consiste à :
- Afficher deux grilles, la grille d'historiques des attaques et la grille des navires (un exemple d'affichage est donné en fin d'énoncé).
- Demander une poition de jeu (A3, B3, etc.).
- Vérifier que la position n'est pas déjà jouée.

Pour un ordinateur, la logique de jeu est à votre convenance.

Un plateau de jeu est composé de deux grilles, une pour conserver l'historique des attaques (*historyBoard*) et la seconde pour voir la position de ses navires et des attaques de l'adversaire (*shipBoard*). Cette dernière grille est constituées de pointeurs sur des navires, par défaut il n'y a que de l'eau, puis les navires du joueur sont placés. Globalement, la classe *BattleShipBoard* contient les fonctions pour afficher, placer et mettre à jour les différentes données concernant les grilles.

Les navires sont tous représentés dans le fichier *ship.cpp*, il en existe cinq types :

Carrier Prend 5 cases.

Battleship Prend 4 cases.

Cruiser Prend 3 cases.

Submarine Prend 2 cases.

Water Navire factice pour représenter l'eau.

Chaque navire retourne un caractère pour représenter son type et être reconnu dans la grille.

Pour finir, le fichier *utils.cpp* propose une fonction utile, et le fichier *definitions.h* quelques définitions.

Question 1 : Compléter le programme

Le programme n'est pas complet car il manque quelques lignes de codes. Le code manquant est identifé par des commentaires dans le code, identifés par la balise *[TODO]*.

Vous devez rendre le répertoire sources contenant vos corrections. Bien entendu, après vos corrections, l'ensemble doit compiler et le jeu doit être possible.

Dans les grandes lignes, voici ce qu'il manque :

- Dans le fichier *game.cpp*, il manque la création des joueurs.
- Dans le fichier *game.cpp*, la condition de victoire n'est pas détectée dans la fonction *play*.
- Dans les fichiers *human.cpp* et *computer.cpp* il manque le code des fonctions *start* et *play*.
- Dans le fichier *board.cpp*, la fonction *reset* n'est pas complète et donc les grilles ne sont pas initialisées.
- Dans le fichier *board.cpp*, la fonction *place* ne fonctionne pas.

Question 2 : Réflexion

Les réponses sont à rendre dans un fichier texte, au nom de l'exercice, dans un des formats suivants : .txt, .odt, .doc(x), .pdf

1. Lorsqu'un navire est coulé, on ne le sait pas à l'écran. Comment pourrait-on améliorer le programme pour prendre en compte cette fonctionnalité ? Proposez une solution plausible.
2. Comment adapter le programme pour joueur à deux joueurs humains ?
3. Si vous deviez écrire des tests unitaires pour la classe *BattleShipBoard*, quelle(s) fonction(s) testeriez-vous ? Quels tests écririez-vous ?

Voici un exemple de jeu en cours. Une attaque dans l'eau est représentée par un X et un navire touché par un O. Sur la grille de droite sont représentés vos navires, et des X pour toutes les attaques de l'ordinateur.

```
    Attack history                          Your Ships
     1   2   3   4   5   6   7   8   9   10      1   2   3   4   5   6   7   8   9   10
   +---+---+---+---+---+---+---+---+---+---+   +---+---+---+---+---+---+---+---+---+---+
A  |   | X |   |   |   |   |   |   | O |   |   | S | X | X | X | X | X | X | . | . | . |
   +---+---+---+---+---+---+---+---+---+---+   +---+---+---+---+---+---+---+---+---+---+
B  |   |   |   |   |   |   |   |   |   |   |   | . | . | C | . | . | . | . | . | . | . |
   +---+---+---+---+---+---+---+---+---+---+   +---+---+---+---+---+---+---+---+---+---+
C  |   |   | O |   |   |   |   |   |   |   |   | . | . | C | . | . | . | . | . | . | . |
   +---+---+---+---+---+---+---+---+---+---+   +---+---+---+---+---+---+---+---+---+---+
D  |   |   | O |   |   |   |   |   |   |   |   | . | . | C | . | . | . | . | . | . | . |
   +---+---+---+---+---+---+---+---+---+---+   +---+---+---+---+---+---+---+---+---+---+
E  |   |   |   |   |   |   |   |   |   |   |   | . | . | C | . | . | R | R | R | . | . |
   +---+---+---+---+---+---+---+---+---+---+   +---+---+---+---+---+---+---+---+---+---+
F  |   |   |   |   |   |   |   |   |   |   |   | . | . | C | . | . | . | . | . | . | . |
   +---+---+---+---+---+---+---+---+---+---+   +---+---+---+---+---+---+---+---+---+---+
G  |   |   |   |   |   |   |   | X |   |   |   | . | . | . | . | . | . | . | . | . | . |
   +---+---+---+---+---+---+---+---+---+---+   +---+---+---+---+---+---+---+---+---+---+
H  |   |   |   |   |   |   |   |   |   |   |   | . | R | . | . | B | B | B | B | . | . |
   +---+---+---+---+---+---+---+---+---+---+   +---+---+---+---+---+---+---+---+---+---+
I  |   |   |   |   |   |   |   |   |   |   |   | . | R | . | . | . | . | . | . | . | . |
   +---+---+---+---+---+---+---+---+---+---+   +---+---+---+---+---+---+---+---+---+---+
J  |   |   |   | X |   |   |   |   |   |   |   | . | R | . | . | . | . | . | . | . | . |
   +---+---+---+---+---+---+---+---+---+---+   +---+---+---+---+---+---+---+---+---+---+
Veuillez entrer une position (ex. E4):
C10
```