### Saé S1.02 : comparaison d'approches algorithmiques

Résolution automatique de Sudoku

### Consignes pour la partie 1

#### Premier programme

Écrivez un premier programme, appelé **ELIMINATION\_Version1.c** basé sur l'algorithme vu en séance de Ressource-SAE.

Vous commencerez par écrire un programme basique qui implémentera uniquement la première technique ("singleton nu"). Testez ce programme avec les différentes grilles fournies sur Moodle.

Puis ajoutez progressivement à votre programme d'autres techniques afin de le rendre de plus en plus performant. Testez-le au fur et à mesure avec les grilles fournies.

Ce premier programme utilisera la première structure de données pour les cases d'une grille de Sudoku, la structure **tCase1** rappelée ici :

```
#define N 3
#define TAILLE (N*N)

typedef struct {
    int valeur;
    int candidats[TAILLE];
    int nbCandidats;
} tCase1;

typedef tCase1 tGrille[TAILLE][TAILLE];
```

#### Deuxième programme

Écrivez une deuxième version de votre programme, appelé **ELIMINATION\_Version2.c** qui utilisera la deuxième structure de données **tCase2** :

```
#define N 3
#define TAILLE (N*N)

typedef struct {
    int valeur;
    bool candidats[TAILLE + 1];
    int nbCandidats;
} tCase2;

typedef tCase2 tGrille[TAILLE][TAILLE];
```

Testez ce deuxième programme avec les grilles fournies sur Moodle.

# Travaux à déposer

Vous déposerez au plus tard le 7 janvier 2024 avant minuit :

- votre premier programme : **ELIMINATION\_Version1.c**, utilisant la structure tCase1
- votre deuxième programme : **ELIMINATION\_Version2.c**, utilisant la structure tCase2
- les captures d'écran des résultats affichés par votre programme pour chaque grille fournie (cf. fichiers .sud). Les résultats devront se présenter sous cette forme :

```
***** RESULTATS POUR Grille99.sud *****

Nombre de cases remplies = 1 sur 58 Taux de remplissage = 1.724 %

Nombre de candidats elimines = 6 Pourcentage d'elimination = 2.597 %
```

### Saé S1.02: comparaison d'approches algorithmiques

Résolution automatique de Sudoku

#### Consignes pour la partie 2

# Première programme: RESOLUTION\_1

Vous devez implémenter en C l'algorithme de backtracking vu en séance de Ressource-SAE, en respectant les consignes suivantes :

- votre programme devra gérer des **grilles 4x4** (constante N=4).
- votre programme devra:
  - lire un fichier de grille de sudoku 4x4<sup>1</sup>,
  - afficher la grille initiale,
  - résoudre la grille,
  - afficher la grille finale (grille complète),
  - afficher le temps CPU de l'étape de résolution.
- vous pourrez bien sûr reprendre des fonctions et procédures de vos programmes antérieurs (pour l'affichage, la vérification, etc.).
- vous testerez ce programme RESOLUTION\_1 avec les grilles 4x4 disponibles sur Moodle<sup>1</sup>.

# Deuxième programme: RESOLUTION\_2

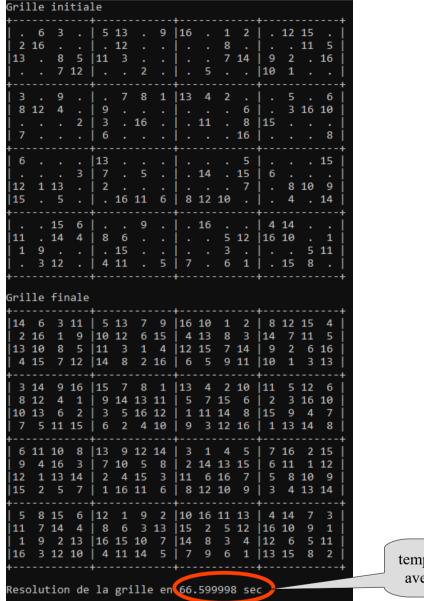
S'il résout une grille de Sudoku à coup sûr, le programme de backtracking est par contre très lent. Réfléchissez à des améliorations possibles et proposez une nouvelle version de votre programme qui soit plus rapide.

Vous testerez ce nouveau programme RESOLUTION\_2 avec les grilles 4x4 disponibles sur Moodle et vous comparerez avec les résultats obtenus précédemment.

### Travaux à déposer

# 1 - Vous déposerez au plus tard le 17 janvier 2024 avant minuit :

- le code source de votre programme de RESOLUTION-1;
- des captures d'écran montrant le temps CPU nécessaire à votre programme pour résoudre chacune des grilles 4x4 fournies. L'affichage prendra cette forme :



temps CPU affiché avec 6 décimales

# 2 - Vous déposerez au plus tard le 28 janvier 2024 avant minuit :

- le code source de votre programme de RESOLUTION-2. Ce nouveau code source devra comporter en <u>commentaire d'entête</u> des explications courtes mais suffisantes sur les améliorations de cette deuxième version du programme.
- comme précédemment, des captures d'écran montrant le temps CPU nécessaire à votre programme pour résoudre chacune des grilles 4x4 fournies (reprendre l'affichage utilisé précédemment).