ATTENTION :
- grille ex. : remplacer un 1 par un 3
- indices tab unidim : N*N (et non N*)
- autre (?) : check

EXAMEN D'INITIATION AUX INTERFACES GRAPHIQUES
Institut Galilée - Université Paris 13 **Mardi 30 avril 2012 – 14h-16h**
Licence 1 **Aucun document autorisé**

Le jeu de Sudoku se joue à un seul joueur sur une grille 9×9, elle-même subdivisée en 9 carrés 3×3. Le but, partant d'une grille contenant quelques chiffres, est de placer un chiffre de 1 à 9 dans chaque case de sorte que :

$\left\{ \begin{array}{l} \text{chacun des 9 carrés} \\ \text{chacune des 9 lignes} \\ \text{chacune des 9 colonnes} \end{array} \right\}$ contienne *exactement une* occurrence de chaque chiffre de 1 à 9. Exemple de grille :

	6			4				9
	3			1		8		2
	4	9	8	3	5			
	1	2			3			
3			7		1			4
			5			1	7	
			3		4	6	8	
6		3		7			2	
1				5			9	

Initialisation

8	6	1	2	4	7	5	3	9
5	3	7	9	1	6	8	4	2
2	4	9	8	3	5	7	1	6
7	1	2	4	6	3	9	5	8
3	5	8	7	9	1	2	6	4
4	9	6	5	8	2	1	7	3
9	7	5	3	2	4	6	8	1
6	8	3	1	7	9	4	2	5
1	2	4	6	5	8	3	9	7

Grille remplie

1 Prise en mains de la bibliothèque "sudoku"

Vous disposez d'une bibliothèque (*cf.*, fichier d'entête joint) représentant une grille de Sudoku. Cette bibliothèque est largement commentée, nous donnons seulement deux précisions complémentaires :

- *Gestion du champ 'nb_coup_joue' et retour arrière.* Le joueur peut placer le chiffre 5 dans la case d'indice $(1, 3)$ au cours du 10è coup, puis le chiffre 9 dans la case $(5, 2)$ au 11è coup, puis changer d'avis sur son coup précédent et placer au 12è coup le chiffre 8 dans la case d'indice $(1, 3)$. À l'issue de ce dernier coup, le champ 'coup_joue' de la case $(1, 3)$ prend la valeur 12, tandis que dans la grille, le champ 'compteur_coup' est inchangé et le champ 'nb_coup_joue' est incrémenté de 1. Supposons que 'nb_coup_joue' vaille alors 12. Si le joueur décide maintenant de faire un retour arrière, la case d'indice $(1, 3)$ est réinialisée et dans la grille, les champs 'compteur_coup' et 'nb_coup_joue' sont décrémentés de 1. Si le joueur opère de nouveau un retour arrière, la case d'indice $(5, 2)$ est réinialisée et les champs 'compteur_coup' et 'nb_coup_joue' sont de nouveau décrémentés de 1. Le champ 'nb_coup_joue' vaut alors 10. Si le joueur opère encore un retour arrière, il n'y a plus dans la grille de case jouée au 10è coup (puisqu'il s'agissait de la case $(1, 3)$, modifiée depuis) : la fonction de retour arrière décrémente alors 'nb_coup_joue', jusqu'à trouver une case jouée au coup "nb_coup_joue', ou jusqu'à ce que 'nb_coup_joue' vale 0.

- *Indice des cases.* Les cases vis-à-vis du joueur ont deux indices x et y, l'un indiquant la ligne (de 1 à 9), l'autre la colonne (également de 1 à 9). Dans la structure de données, les cases sont stockées dans un tableau unidimensionnel, elles n'ont donc qu'un indice. La case d'indice (x, y) dans le jeu est représentée dans la structure par la case d'indice $(N * N * (x - 1)) + (y - 1)$ dans le tableau 'tab_case'.

1.1 Prise en mains des structures de données & de la bibliothèque "sudoku"

1. Suite à une mauvaise manipulation, il faut réécrire certaines fonctions de la bibliothèque "sudoku". Il s'agit des fonctions `case_get_val_joue`, `case_set`, `grille_initialiser` et `grille_est_resolue`.

2. Il manque certaines fonctionnalités à la bibliothèque "sudoku". Donner le prototype et écrire le code de deux nouvelles fonctions :

   (a) La fonction `grille_afficher` qui représente l'état de la grille, comme dans les figures du haut.

   (b) La fonction `grille_derouler` qui permet de retracer le déroulement du jeu en affichant, pour chaque coup joué de 1 au nombre de coups joués de la partie : si une case a été jouée lors de ce coup (et n'a pas été rejouée depuis), l'indice de ligne de cette case, l'indice de colonne de cette case, et la valeur

attribuée. Ex. d'affichage :

```
coup 1:  1  1  2
coup 2:  -  -  -
coup 3:  1  3  8
coup 4:  2  1  7
coup 5:  2  3  5
```

3. Donner, en justifiant pas à pas leur évolution, la valeur des champs 'compteur_joue' et 'nb_coup_joue' de la grille à la suite de la succession suivante d'instructions :

```
1   grille_sudoku une_grille;
2   grille_initialiser(& une_grille);
3   grille_joueurcoup(& une_grille, 1, 1, 2);
4   grille_joueurcoup(& une_grille, 1, 3, 5);
5   grille_joueurcoup(& une_grille, 1, 3, 8);
6   grille_joueurcoup(& une_grille, 2, 1, 7);
7   grille_joueurcoup(& une_grille, 2, 3, 5);
8   grille_dejoueurcoup(& une_grille);
9   grille_dejoueurcoup(& une_grille);
10  grille_dejoueurcoup(& une_grille);
11  grille_dejoueurcoup(& une_grille);
12  grille_dejoueurcoup(& une_grille);
```

2 Interface graphique du jeu de Sudoku

Vous devez développer l'interface graphique du jeu de sudoku, en gtk. L'interface attendue est minimaliste : lorsqu'on lance l'application, une grille s'affiche dans une fenêtre. Cette grille doit permettre au joueur d'inscrire un chiffre dans une case, et ce tant que la grille n'est pas totalement remplie. Le nombre de chiffres qui restent à définir est indiqué par le titre de la fenêtre, qui est de la forme "Sudoku : xx valeur(s) à définir". Enfin un bouton "Rejouer" permet de renouveler la grille.

1. Composants graphiques (**NB** : vous pouvez évidemment répondre en vous aidant d'un schéma) :
   (a) Quels sont les objets graphiques impliqués (*ex.*, 3 boutons, 2 libellés, etc.) ?
   (b) Comment ces objets sont-ils imbriqués entre-eux (*ex.*, le bouton `toto` dans la fenêtre `tata`, *etc.*) ?
   (c) Pour chaque imbrication d'un objet dans un autre, indiquer et justifier la fonction gtk à utiliser. Vous pouvez pour cela donner le prototype de la fonction si vous le connaissez, ou simplement décrire la fonction, l'objet gtk sur lequel elle est définie, sa vocation, ses paramètres essentiels.
2. Leur comportement :
   (a) Indiquer pour chaque composant interactif (*ex.* bouton, zone de saisie, *etc.*) son état d'activité en fonction de l'état de la grille.
   (b) Indiquer pour chaque libellé (qu'il s'agisse du titre d'une fenêtre, du contenu d'une zone de saisie, de l'icône d'un bouton, *etc.*) l'évolution de sa valeur en fonction de l'état de la partie.
3. Sollicitations faites au modèle (**NB** : pour les fonctions gtk, il n'est pas nécessaire d'en connaître le nom exact, mais leur vocation, et le type des arguments auxquels elles s'appliquent) :
   (a) Par quelle fonction lance-t-on une interface graphique gtk ?
   (b) Quelle(s) fonction(s) de la bibliothèque "sudoku" faut-il appeler préalablement au lancement de l'interface graphique ?
   (c) Quelle(s) fonction(s) de la bibliothèque "sudoku" faut-il appeler lorsque sur l'interface, le joueur a choisi d'attribuer une valeur donnée à une case donnée ?
   (d) Une fois une interface gtk lancée, comment peut-on l'arrêter : par quelle action de l'utilisateur, par quelles fonctions gtk, par quels appels faits à ces fonctions dans votre programme ?
4. Comment faire en sorte que votre interface ait le comportement décrit dans les réponses précédentes ? Indiquer :
   (a) le prototype et la vocation des fonctions que vous envisagez devoir écrire (mais sans écrire les fonctions) ;

(b) l'ensemble des appels que vous envisagez de faire à la fonction `gtk_signal_connect` ;

(c) les éventuelles structures de données que vous envisagez de définir pour faire appel à ces fonctions.