Le mini-cactpot est un mini-jeu de hasard qu'on retrouve dans le jeu Final Fantasy XIV.

Le mini-cactpot est une grille de neuf cases dans lesquelles il y a un chiffre entre 1 et 9. Au début, une seule case est affichée. On dispose ensuite de 3 cases supplémentaires qu'on peut dévoiler. Une fois toutes les cases possibles dévoilées, on choisit une colonne, une ligne, ou une diagonale, et selon la somme de tous les chiffres on obtient une certaine récompense.

On va recréer ici une page web pour jouer à ce mini-jeu.

QUESTION 0

Prenez un temps pour lire les fichiers `cactpot.html` et `cactpot.js`.

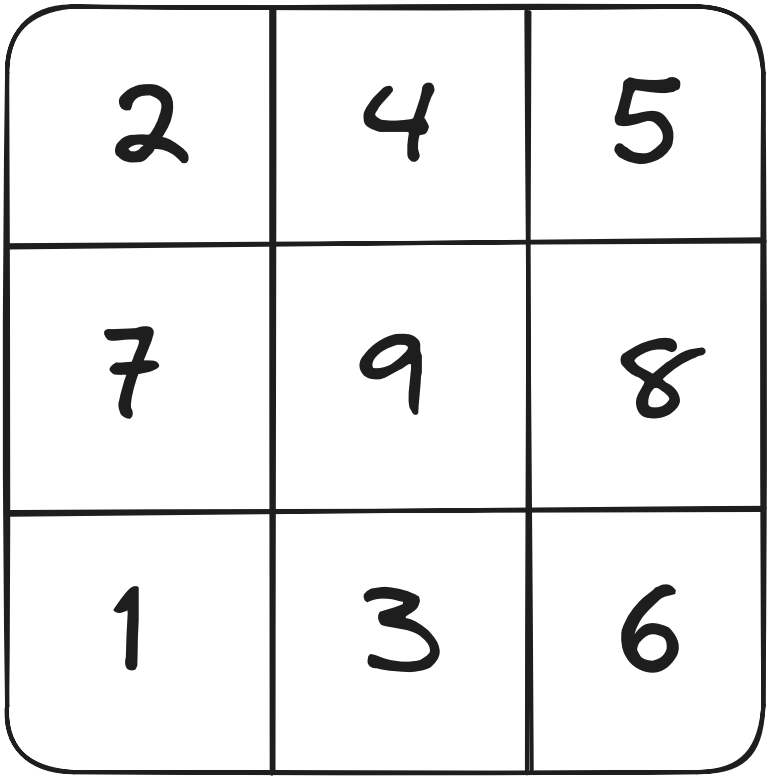
QUESTION 1

On vous fournit la fonction `valeurs_tableau()` qui renvoie un `Array` avec des chiffres de 1 à 9, triés aléatoirement, et non-répétés.

Dans cette question, vous devrez implémenter la fonction `remplir_table` qui prend en argument un tableau créé par la fonction `valeurs_tableau()`.
Il vous faut :

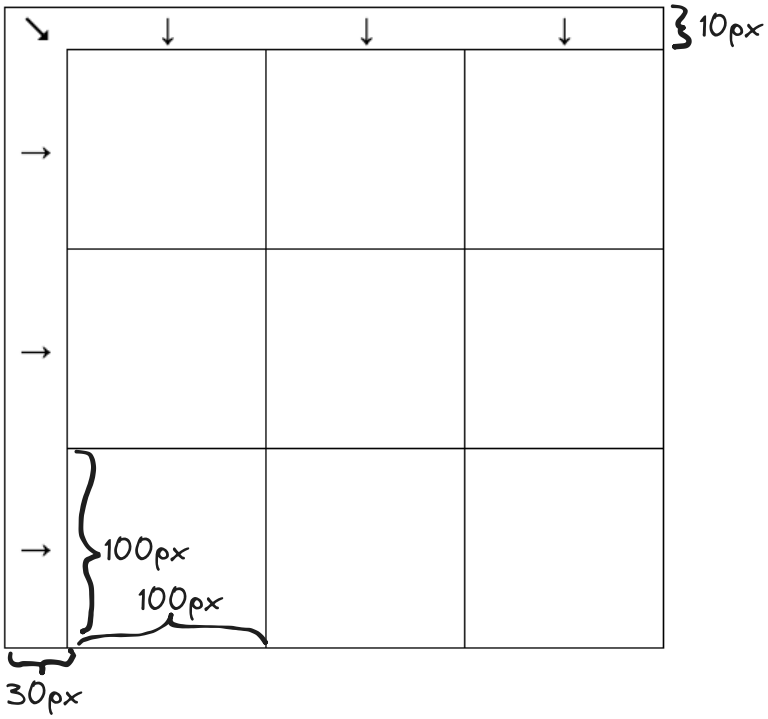
- récupérer la liste de tous les éléments `td` ;
- pour chaque élément, remplissez son `innerHTML` avec la valeur qui lui correspond dans le tableau.

Par exemple, si on a l'array `[2, 4, 5, 7, 9, 8, 1, 3, 6]` on devra retrouver :



QUESTION 2

Modifiez le fichier CSS pour que votre page ressemble à quelque chose comme ceci :



Cherchez dans le fichier HTML les identifiants et/ou les noms de class qui correspondent aux éléments à modifier, et modifiez-les.

QUESTION 3

On veut pouvoir récupérer la liste des identifiants des `td` qui sont sur le chemin d'une des flèches. Pour cela, on va implémenter la fonction `ids_correspondants` qui prend en argument l'identifiant d'une des flèches, et qui renvoie l'identifiants des cellules qui lui correspondent.

Par exemple,

- `ids_correspondants("l1")` doit renvoyer `["11", "12", "13"]` ;
- `ids_correspondants("c2")` doit renvoyer `["12", "22", "32"]` ;
- `ids_correspondants("d1")` doit renvoyer `["11", "22", "33"]`.

> **Indice :**
>
> Rappelez-vous qu'une chaîne de caractère est simplement un tableau de caractère. Par exemple, si j'ai `let str = "l1"` alors `str[0]` vaut `"l"` et `str[1]` vaut `"1"`.

QUESTION 4

Implémentez maintenant la fonction `somme_cases` qui prend en argument une liste d'identifiants d'élément `td` comme renvoyée par la fonction `ids_correspondants`, et qui renvoie la somme des valeurs dans les `td`.

> **Rappel**
>
> Pour transformer un caractère en entier, il existe la fonction `parseInt`.

QUESTION 5

Implémentez la fonction `afficher_cases` qui prend en argument une liste d'identifiants d'éléments `td` comme renvoyée par `ids_correspondants`, et qui modifie le style de chaque `td` de la façon suivante :

- le fond des cases est gris (ou la couleur de votre choix) ;
- le chiffre est écrit en noir.

QUESTION 6

Dans la fonction `init_page` vous pouvez voir que quand on clique sur un `<th>`, la fonction `th_click_event` est appelée.

Cette fonction doit :

- récupérer avec `ids_correspondants` la liste des id des `<td>` qui sont sélectionnés par le `th` cliqué ;
- afficher les cases des `<td>` ;
- afficher dans la console la somme calculée par `somme_cases`.

QUESTION 7

L'utilisateur n'a le droit qu'à 3 clicks pour révéler le contenu d'une cellule.

Pour l'instant, toutes les cellules sont visibles, mais on va quand même s'occuper de compter les clicks maintenant.

Pour implémenter la fonction `decrementer_coups_restants()`, il vous faudra :

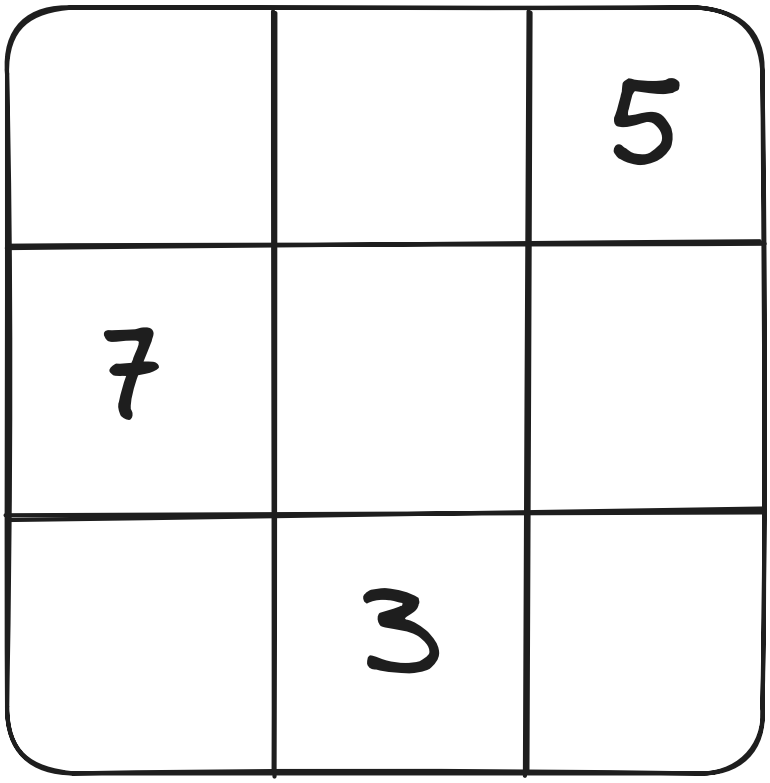
1. Retrouver dans le HTML la partie dédiée à afficher les coups restants ;
2. Sachant que le nombre de click effectués est stocké dans la variable `nb_click_td`, tester sa valeur, de façon à ce que:
   - s'il reste encore des coups, afficher le nombre de coups restants ;
   - sinon, afficher "Plus de coups restants. Cliquez sur une flèche.".
3. Changer le contenu du bon élément HTML.

QUESTION 8

Au début de la partie, seules 3 cases, choisies aléatoirement, sont visibles.

Implémentez la fonction `cacher_afficher_cases(cases_affichees)` qui récupère toutes les cellules `<td>` et qui change leur couleur en blanc si sa position dans la liste des `<td>` n'est pas présente dans `cases_affichees` passé en argument.

Par exemple, `cacher_afficher_cases([3, 4, 8])` sur l'exemple de la question 1 affichera :



QUESTION 9

Quand on clique sur une cellule `<td>`, la fonction `afficher_td` est appelée.

Cette fonction doit :

- changer la couleur de la cellule cliquée en noir ;
- incrémenter `nb_click_td` ;
- appeler `decrementer_coups_restants()` pour mettre à jour l'affichage du nombre de coups restants ;
- s'il n'y a plus de coups restants, on récupère toutes les cellules `td` et on enlève l'écouteur avec `td.removeEventListener("click", afficher_td)` afin d'empêche la personne qui joue de révéler plus de cases.

QUESTION 10

Le jeu est fini quand on a cliqué sur une des flèches. À ce moment-là, il faut afficher les cellules qu'il faut, et qu'on affiche le résultat.

Pour cela, vous devrez :

- récupérer tous les `<td>` ;
- les rendre tous visibles en remettant leur couleur en noir ;
- récupérer tous les `<th>` ;

- pour chaque `<th>`, enlever l'écouteur sur click qu'il y avait dessus avec `th.removeEventListener("click", th_click_event)` ;
- pour chaque `<div>` dont l'attribut `class` vaut `cacher`, le rendre visible ;
- afficher la chaîne `str` au bon endroit ;
- dans `th_click_event`, appeler `finir_jeu` avec le résultat de la somme.

QUESTION 11

Pour pouvoir rejouer sans actualiser la page, on propose à la fin de chaque partie un bouton qui initialise une nouvelle partie.

Quand on clique sur ce bouton, la fonction `click_bouton_rejouer()`, qui doit :

- pour chaque `<div>` qui était cachée avant la fin de la partie, on la recache ;
- appeler la fonction `init_page()`.