Fiche annexe sur la mise en echec.

Fonction de détection

Les fonctions de gestions des conditions de mises en echec ainsi que de victoires/défaites seront regroupés dans un fichier distinct.
On créera pour cela deux nouveaux fichiers: *algorithmique_partie.c* et *algorithmique_partie.h*

On créera la fonction *algorithmique_partie_tester_mise_en_echec* dont le contrat et la signature seront les suivants:

```
/**
 * Fonction mise_en_echec_tester:
 * ******************************
 *   Recherche si le joueur courant est mis en echec dans la configuration
courante.
 *
 *   Necessite:
 *     - Un pointeur constant vers un echiquier (pointeur non NULL).
 *   Garantie:
 *     - Renvoie 1 si le joueur courant est mis en echec par le joueur adverse.
 *     - Renvoie 0 sinon.
 */
int algorithmique_partie_tester_mise_en_echec(const echiquier*
echiquier_courant);
```

L'algorithme pourra être le suivant:

```
    //Algorithme
    // Pour toutes les pieces attaquantes
    //   Si deplacement standard de piece attaquante vers piece courante est
valide
    //      Alors retourner 1
    // retourner 0
```

Notes:
- Il sera nécessaire de travailler sur un échiquier où le joueur pouvant déplacer des pièces correspond au joueur adverse.
- N'oubliez pas d'utiliser les fonctions de hauts niveaux codées tout au long du projet pour éviter de manipuler les champs des structures directement (*echiquier_changer_joueur_courant*, *jeu_recuperer_roi_information*, etc)

Intégration dans l'API

La mise en echec possède deux actions distinctes:
1. L'indication de la mise en echec du joueur adverse après déplacement d'une pièce.
2. L'invalidation de la demande d'un déplacement de pièce car celle-ci viendrait laisser le roi en echec.

Le premier cas sera géré uniquement visuellement au niveau de l'API dans la fonction *api_echec_deplacer_piece.*
Lorsqu'un coup est joué, après changement de joueur, vérifiez si le nouveau joueur est mis en echec et indiquez: "Joueur X en echec" dans ce cas (avec X valant 0 ou 1).

Le second cas est traité au niveau du déplacement des pièces dans la fonction *echiquier_deplacement_realise_si_possible.*
Si un déplacement est considéré comme valide, avant d'écrire le contenu de l'échiquier temporaire dans l'échiquier courant, vérifiez que ce coup ne laisse pas le joueur courant en echec.
Si tel est le cas, invalidez le coup en ne transférant pas le contenu de l'échiquier temporaire et en plaçant le code d'erreur à *erreur_roi_en_echec*, ainsi que le type de coup à *deplacement_invalide*.

Note. Référez-vous également à l'annexe concernant l'organisation générale des déplacements.