Sujet 3:

Mise en place du mouvement de l'action de changement d'orientation.

(durée estimée: 1h)

➔ A partir du contrat donné dans `api_siam_tenter_changer_orientation_piece_si_possible`, complétez la fonction en vous servant des deux précédentes (`plateau_changer_orientation_piece`) ainsi que des vérifications supplémentaires.

➔ Vérifiez que votre code est fonctionnel de bout en bout. Pour cela, vous pouvez tenter de changer l'orientation d'une pièce à l'aide du mode interactif (appelez celui-ci depuis votre fonction main après avoir initialisé un jeu avec des pièces déjà présentes sur le plateau (vous pourrez temporairement modifier la fonction jeu_initialiser())). Notez qu'il est important de vérifier que les coups valides sont bien réalisés, mais que les coups invalides sont également reconnus comme tels. Et ceux-ci ne doivent pas faire planter le programme.

Rem. Si vous lancez l'executable depuis QtCreator et non depuis le terminal, il est nécessaire de configurer le lancement en mode "terminal" afin de pouvoir interagir avec le mode interactif au clavier.

Mise en place des autres coups.

(durée estimée: 2h30min)

➔ Implémentez le reste des coups: introduction d'une nouvelle pièce et déplacement d'une pièce existante sans considérer l'action de poussée (pour l'instant, tout coup nécessitant une poussée sera considéré comme invalide). N'oubliez pas d'écrire dans le corps des fonctions l'algorithme que vous suivez en commentaire. N'oubliez pas non plus de mettre à jour les fonctions de l'API afin de rendre votre jeu fonctionnel depuis le mode interactif.

Gestion de la poussée.

(durée estimée: 30min)

Nous allons regrouper ensemble les fonctions gérant la poussée. Pour cela, nous définissons les deux fonctions suivantes données par ces contrats:

```
/**
 * Fonction poussee_etre_valide:
 * ***************************
 *    Verifie si il est possible de realiser une poussee qui debute
 *     aux coordonnees (x,y) dans l'orientation definie.
 *    Note: Les coordonnees (x,y) designent la premiere piece rencontree
 *     dans la poussee.
 *
 *    Necessite:
 *      - Un pointeur non NULL vers un plateau integre non modifiable.
 *      - Deux coordonnees entieres (x,y) designant une piece non
 *         vide integre du plateau.
 *      - Une orientation de poussee designant une direction integre.
 *    Garantie:
 *      - Un retour valant 1 si la poussee est possible.
 *      - Un retour valant 0 sinon.
 *
 */

/**
 * Fonction poussee_realiser:
 * ************************
 *    Realise une poussee sur un plateau en supposant que celle-ci
 *     est realisable.
 *
 *    Necessite:
 *      - Un pointeur non NULL vers un plateau integre modifiable.
 *      - Deux coordonnees entieres (x,y) designant une piece non
 *         vide integre du plateau.
 *      - Un type d'animal a deplacer.
 *      - Une orientation de poussee designant une direction integre.
 *      - Un pointeur non NULL vers une condition de victoire modifiable.
 *    Garantie:
 *      - Un plateau integre mis a jour en ayant realise la poussee.
 *
 */
```

➔ Créez deux nouveaux fichiers: `poussee.h` et `pousse.c` qui géreront ces deux fonctions.

Placez l'en-tête dans le fichier .h sans chercher à complétez pour l'instant le corps des fonctions dans le fichier .c (vous pouvez complétez ces fonction de manière arbitraire par exemple avec un `return 1;` pour la fonction renvoyant un entier.). Complétez votre script de compilation afin de prendre en compte ces nouveaux fichiers.

Travail en autonomie

(durée estimée: 3h):

- ➔ **Ecrire** l'algorithme du corps des fonctions: `poussee_etre_valide` et `poussee_realiser`.
- ➔ **Lire** la documentation sur la compilation pour la prochaine séance.
- ➔ **Faire et répondre** aux questions du tutoriel sur les étapes du préprocesseur et de compilation.

Pour cela, vous devez télécharger l'archive contenant les tutoriaux d'entraînements.
Les programmes correspondants aux préprocesseurs et à la compilation sont dans les répertoires 01_preprocesseur, et 02_compilation.
Pour chaque répertoire, il y a 3 niveaux d'exercices:

- *a_minimal*: A faire obligatoirement avant la semaine suivante, ce sont les bases minimales à maîtriser absolument avant la séance suivante.
- *b_recommandé*: Notions recommandées à connaître. Couvre différents points qui sont supposés être maîtrisés pour le partiel final.
- *c_avancé*: Notions avancées pour les étudiants souhaitant aller plus loin. Ces notions ne seront pas exigibles pour le partiel.

Note: Les réponses ne seront pas ramassées: posez des questions si vous n'êtes pas sûr. Les points couverts se retrouveront dans le partiel écrit.