MISSION 2 : CHECKLIST – BRANCHES & CONFLITS

Objectif : comprendre, créer, utiliser et fusionner des branches, puis gérer un vrai conflit Git.

Étape 1 — Préparation de la branche principale (main)

⬜ Ouvrir le projet dans VS Code

⬜ Vérifier la branche actuelle : `git branch`

⬜ Vérifier qu'il n'y a aucune modification non committée : `git status`

⬜ Faire un commit "checkpoint" si nécessaire :

```
git add .
git commit -m "Checkpoint avant création de branche"
```

Étape 2 — Créer une nouvelle branche

Objectif : isoler une nouvelle fonctionnalité dans sa propre branche.

⬜ Créer une branche nommée **fonction_1** :

```
git branch fonction_1
```

⬜ Vérifier la liste des branches :

```
git branch
```

⬜ Se déplacer sur la nouvelle branche :

```
git checkout fonction_1
```

⬜ Vérifier que la branche active est bien fonction_1

Étape 3 — Travailler dans la branche fonction_1

Objectif : créer un vrai commit dans la nouvelle branche.

⬜ Modifier un fichier (README ou fichier du projet)

⬜ `git status`

⬜ Ajouter le fichier : `git add nom-du-fichier`

⬜ Créer un commit clair : `git commit -m "Ajout d'une fonctionnalité dans fonction_1"`

⬜ Confirmer que le commit est bien dans fonction_1 : git log --oneline --decorate

Étape 4 — Revenir sur main et créer un changement différent

⬜ Revenir sur la branche principale :

```
git checkout main
```

⬜ Modifier **le même fichier**, mais d'une manière différente (phrase différente, ajout différent)

⬜ Ajouter et committer :

```
git add nom-du-fichier
git commit -m "Modification dans main"
```

⬜ Vérifier que main et fonction_1 ont désormais des commits différents

```
git log --oneline --graph --decorate --all
```

Étape 5 — Fusionner fonction_1 dans main (cas avec conflit)

⬜ Être sur la branche main :

```
git checkout main
```

⬜ Lancer la fusion :

```
git merge fonction_1
```

⬜ Observer l'apparition du conflit : Git indique le fichier en conflit.

⬜ Ouvrir le fichier concerné

⬜ Repérer les marqueurs :

```
<<<<<<< HEAD
(texte de main)
=======
(texte de fonction_1)
>>>>>>> fonction_1
```

Étape 6 — Résoudre le conflit

⬜ Choisir le texte à conserver :

- celui de main
- celui de fonction_1
- un mélange des deux

⬜ Supprimer tous les marqueurs :

```
<<<<<<<
=======
>>>>>>>
```

⬜ Sauvegarder le fichier

⬜ Ajouter le fichier résolu :

```
git add nom-du-fichier
```

⬜ Finaliser la fusion :

```
git commit -m "Résolution du conflit entre main et fonction_1"
```

Étape 7 — Vérification de la fusion

⬜ Afficher le graphe :

```
git log --oneline --graph --decorate --all
```

⬜ Vérifier que :

- les deux branches sont fusionnées
- le commit de merge est présent
- le fichier contient bien le texte final choisi

Étape 8 — Nettoyage de fin de mission

⬜ Supprimer la branche fonction_1 (si elle n'est plus utile) :

```
git branch -d fonction_1
```

⬜ Vérifier la liste des branches : `git branch`

⬜ S'assurer qu'il n'y a plus de conflit ni de modifications en attente : git status

MISSION VALIDÉE SI :

- Une branche **fonction_1** a été créée
- Un commit a été fait dedans
- Le même fichier a été modifié différemment sur main
- Un conflit a été déclenché lors du merge
- Le conflit a été résolu proprement
- Un commit de fusion est présent
- Le graphe est propre et complet
- La branche fonction_1 a été supprimée (optionnel mais recommandé)