Plan de codage — Partie 1 (Répartition par membre)

Étape	Membre 1 — main.c	Membre 2 — adj_list.c / .h	Membre 3 — markov_check.c / .h	Membre 4 — export_mermaid.c / .h
1. Initialisation	Configure CMakeLists.txt (cible principale) et inclusions.	Crée adj_list.h avec structures de base.	Crée markov_check.h avec prototypes de fonctions.	Crée export_mermaid.h avec prototypes et constantes Mermaid.
2. Structures & prototypes		typedef struct Edge {...}; typedef struct {...} AdjList; Implémente adj_create, adj_add, adj_print, adj_free, adj_read_file.	Définit markov_is_valid() et markov_report().	Définit node_id() et write_mermaid().
3. Lecture et construction du graphe		Implémente adj_read_file() : lit fichier data/input.txt, n, u v p, gère indices 1→0, erreurs.		
4. Affichage du graphe		adj_print() : affiche chaque sommet et ses arêtes (u -> v, p).		
5. Vérification Markov	Appelle markov_report() et affiche résultat global.	Fournit la structure AdjList en paramètre.	Implémente markov_is_valid() et markov_report() avec tolérance [0.99;1.00].	
6. Export Mermaid (si valide)	Si markov_is_valid() == 1 → appelle write_mermaid().			Implémente node_id() (A, B, C… AA, AB…) et write_mermaid() (flowchart LR + A -->|p| B).
7. Nettoyage mémoire	Appelle adj_free() avant fin du programme.	Implémente adj_free() pour chaque liste et tableau.		
8. Gestion erreurs / retours	Codes retour : 0=OK, 1=non-Markov, 2=erreur fichier/allocation.	Messages d'erreur lecture.	Affiche valeurs hors bornes et nœuds fautifs.	Affiche erreur écriture fichier.
9. Tests et validation	Test data/valid.txt, borderline.txt, invalid.txt.	Vérifie affichage correct.	Vérifie détection non-Markov.	Vérifie rendu Mermaid sur mermaid.live.
10. Livraison finale	Produit exécutable graph_part1 et README final.	Code commenté et sans warnings.	Messages propres et précis.	Fichier .mmd valide visuellement.