Gestion des Risques et des Rendus

Equipe GL 13

Gestion des Risques

Identification des risques

- **Risque technique : Partie B.** Légère approche orientée objet.
- **Risque technique : Partie C.** Faiblesses générales, erreurs dans les opérations arithmétiques et sémantique dynamique de Deca.
- **Risque organisationnel.** Retards possibles dus aux corrections nécessaires pour la partie sans objet.
- **Risque de qualité.** Faible couverture de tests pour les parties B et C, corrections en cours et nombreux internal compiler error.
- **Risque lié aux outils.** Aucun problème détecté à ce jour.

Évaluation des risques

Chaque risque est évalué sur deux critères principaux : **Probabilité** (faible, moyenne, élevée) et **Impact** (faible, moyen, élevé). Les risques techniques et de qualité sont prioritaires, car ils affectent directement la fiabilité et la fonctionnalité du compilateur.

Stratégies pour la suite

- **Risque technique** :
  - Renforcer les tests unitaires et d'intégration pour les parties critiques.
- **Risque organisationnel** :
  - Répartir les tâches liées à la partie sans objet sur plusieurs membres pour accélérer les corrections.
  - Suivre une révision journalière des avancements, continuer avec le rythme soutenu des derniers jours.
- **Risque de qualité** :
  - Établir un plan de tests exhaustif pour les scénarios critiques.
  - Effectuer des revues régulières des tests pour garantir une couverture optimale.
- **Risque lié aux outils** :
  - Mettre en place une documentation claire pour les outils et les scripts utilisés.

Plan d'urgence

En cas de réalisation d'un risque majeur :

- Réorienter les priorités sur les fonctionnalités essentielles et les tester.
- Allouer du temps supplémentaire pour les corrections critiques.

Gestion des Rendus

Organisation des livrables

- **Rendu intermédiaire** : Implémentation complète des fonctionnalités de base, tests minimaux et documentation initiale.
- **Rendu final** : Compilateur complet avec fonctionnalités avancées, couverture de tests optimisée, et guide utilisateur détaillé.
- **Documentation** : Manuel utilisateur, description des algorithmes et des choix techniques, et instructions pour les tests.

Procédures de tests avant les rendus

- Utilisation de scripts automatisés pour valider les fonctionnalités.
- Analyse détaillée des erreurs signalées et correction immédiate.
- Tests organisés en catégories (valide, invalide).

Contrôles de qualité

- Respect strict des spécifications, avec vérification des comportements en ligne de commande.

Gestion des versions

- Maintenir une branche `master` stable, avec des tests vérifiés avant tout merge.
- Travailler sur des branches dédiées pour les nouvelles fonctionnalités (`feature/*`).

Processus d'intégration continue

- Automatisation des tests et vérifications après chaque modification.

Anticipation des pénalités

- Réserver une période tampon pour résoudre les éventuels échecs des tests critiques.
- Établir une checklist finale pour garantir la conformité avant chaque soumission.