Projet : simuler des manifestations

1 Contexte

Une manifestation est une réunion organisée sur la voie publique dans le but d'exprimer une conviction collective [1]. Elle prend le plus souvent la forme d'un cortège qui se déplace. Un cahier des charges est donné ci-après dans le cadre de ce projet.

- Un **cortège** est constitué de plusieurs groupes de personnes. Un cortège a un nom qui exprime le sujet de la manifestation.
- Un **groupe** de personnes est constitué de plusieurs personnes rangées dans leur ordre d'arrivée au sein du groupe. La personne la plus ancienne est le leader du groupe. Un groupe possède un nom et une couleur.
- Une **personne** possède un nom, un identifiant unique et une position. Les identifiants sont des entiers générés automatiquement.
- Une **manifestation** consiste à déplacer un cortège sur une route droite. Une route est représentée par une grille de positions de la forme (x, y) avec $1 \leq x \leq$ largeur et $1 \leq y \leq$ longueur où la largeur et la longueur sont paramétrables.
- Les groupes dans un cortège peuvent être triés par ordre alphabétique des couleurs ou par ordre décroissant des tailles.
- Le déplacement est illustré sur l'exemple ci-après.

On spécifie la manifestation dans un fichier de la forme suivante. Le sujet de la manifestation est sur la première ligne. Chaque autre ligne donne le nom d'un groupe, sa couleur et sa taille.

```
Mu Zik !
rock;bleu;6
hip-hop;magenta;5
electro;vert;8
```

On génère automatiquement les groupes du cortège à partir de ce fichier et en puisant les noms des personnes aléatoirement dans un fichier des prénoms de l'INSEE « `nat2021.csv` ».

- rock : YESIM, JOREN, HERIZO, LOETITIA, CHARLAINE, LUNA-MARIA
- hip-hop : MICHAELLE, JOWAN, ELVEN, GWENEL, DÉBORA
- electro : GUADALUPE, DORIENNE, JUN, LÉANDRA, LIHA, ROBINE, ROKHIA, SADJA

On donne un exemple de déplacement sur une grille de largeur 5 et de longueur 7. Au début de la manifestation, la grille est vide. La position de chaque personne est égale à $(0, 0)$.

7	–	–	–	–	–
6	–	–	–	–	–
5	–	–	–	–	–
4	–	–	–	–	–
3	–	–	–	–	–
2	–	–	–	–	–
1	–	–	–	–	–
	1	2	3	4	5

1. Voir `https://www.vie-publique.fr`

On simule une étape. Les cinq personnes en tête du cortège s'avancent sur la première ligne. On représente ci-dessous leurs initiales dans la couleur du groupe. Il s'agit de YESIM, JOREN, HERIZO, LOETITIA, CHARLAINE du groupe rock.

7	–	–	–	–	–
6	–	–	–	–	–
5	–	–	–	–	–
4	–	–	–	–	–
3	–	–	–	–	–
2	–	–	–	–	–
1	*Y*	*J*	*H*	*L*	*C*
	1	2	3	4	5

On simule une seconde étape. Les personnes présentes sur la grille avancent toutes d'un pas en avant. Puis les cinq personnes suivantes du cortège s'avancent sur la première ligne. Il s'agit de LUNA-MARIA du groupe rock et de MICHAELLE, JOWAN, ELVEN, GWENEL du groupe hip-hop.

7	–	–	–	–	–
6	–	–	–	–	–
5	–	–	–	–	–
4	–	–	–	–	–
3	–	–	–	–	–
2	*Y*	*J*	*H*	*L*	*C*
1	*L*	*M*	*J*	*E*	*G*
	1	2	3	4	5

On simule la troisième et la quatrième étapes de la même façon.

7	–	–	–	–	–
6	–	–	–	–	–
5	–	–	–	–	–
4	–	–	–	–	–
3	*Y*	*J*	*H*	*L*	*C*
2	*L*	*M*	*J*	*E*	*G*
1	*D*	*G*	*D*	*J*	*L*
	1	2	3	4	5

7	–	–	–	–	–
6	–	–	–	–	–
5	–	–	–	–	–
4	*Y*	*J*	*H*	*L*	*C*
3	*L*	*M*	*J*	*E*	*G*
2	*D*	*G*	*D*	*J*	*L*
1	*L*	*R*	*R*	*S*	–
	1	2	3	4	5

Lors d'une étape, les personnes sur la dernière ligne (la 7 ici) sortent de la manifestation. On peut donc simuler une manifestation jusqu'à vider la grille.

On s'intéresse également au repérage et à l'extraction d'une personne connaissant son identifiant, en cas d'urgence par exemple. Dans la grille ci-dessus à droite, on peut extraire par exemple JOWAN du groupe hip-hop. Il en découle un décalage de toutes les personnes suivantes, comme suit. Pour ne pas laisser d'espace vide, s'il reste au moins une personne qui n'a pas commencé à défiler alors elle rentre sur la position (largeur, 1).

7	–	–	–	–	–
6	–	–	–	–	–
5	–	–	–	–	–
4	*Y*	*J*	*H*	*L*	*C*
3	*L*	*M*	*E*	*G*	*D*
2	*G*	*D*	*J*	*L*	*L*
1	*R*	*R*	*S*	–	–
	1	2	3	4	5

La position (x, y) d'une personne extraite renseigne sur l'endroit où elle se trouve : au début de la manifestation si $y = 0$, à la fin si $y >$ longueur ou au niveau de la ligne y sinon.

2 Travail

Vous devrez commencer par récupérer le fichier `manif_etu.zip` sur `madoc`. Le travail est organisé en trois phases décrites ci-après.

L'instruction `make` permet de compiler le code. Il est fortement conseillé de réutiliser les structures de la librairie standard pour votre implémentation.

2.1 Groupe

Un groupe est une collection de personnes rangées de la plus ancienne (la première) à la plus récente (la dernière), muni des opérations suivantes dans la SDA : création d'un groupe vide, destruction, taille, accès à une personne à partir de son identifiant, insertion d'une personne, suppression (extraction) d'une personne à partir de son identifiant, suppression de la première personne, accès au leader. La complexité temporelle de toutes ces opérations doit être un $\Theta(1)$.

Un groupe doit être itérable, c'est-à-dire qu'il doit fournir un type d'itérateur pour accéder aux personnes, ainsi que les méthodes `begin` et `end`.

Vous devrez suivre une méthode de développement raisonnable : réfléchir avant de coder ; coder un peu et tester beaucoup ; mettre des assertions pour vérifier les préconditions ; commenter les parties non triviales du code, les attributs, etc.

2.2 Cortège

Un cortège est une liste de groupes, muni des opérations suivantes dans la SDA : création d'un cortège vide, destruction, insertion d'un groupe, suppression d'un groupe à partir de son nom, accès à une personne à partir de son identifiant, suppression (extraction) d'une personne à partir de son identifiant, tri par couleurs des groupes, tri par tailles des groupes.

Un cortège doit être itérable, c'est-à-dire qu'il doit fournir un type d'itérateur pour accéder aux groupes, ainsi que les méthodes `begin` et `end`.

2.3 Manif

Une manif est un cortège qui se déplace, comme décrit précédemment, munie des opérations suivantes dans la SDA : création, destruction, simulation d'une étape, test de fin, accès à une personne à partir de son identifiant, suppression d'une personne à partir de son identifiant, accès à l'ensemble des leaders en train de défiler.

3 Consignes

Les consignes sont des impératifs à respecter.

3.1 Méthode de travail

Le travail se fait par binômes formés au sein d'un même groupe de TD. Vous devrez inscrire votre binôme au plus tard le **lundi 27 février à 12h** dans le fichier dédié sur `madoc`. Si votre binôme n'est pas inscrit dans ce fichier à cette date, votre note finale sera 0. En cas de problème pour trouver un binôme, il faudra vous identifier auprès de votre chargé de TD/TP le plus tôt possible, et ce au plus tard le 27 février.

Il est conseillé de créer un dépôt `git` pour travailler de manière collaborative.

Comme dit ci-avant, vous devrez suivre une méthode de développement raisonnable : réfléchir avant de coder ; définir des SDA claires ; choisir des SDC pour avoir les meilleures complexités possibles des algorithmes ; coder un peu et tester beaucoup ; mettre des assertions pour vérifier les préconditions ; commenter les parties non triviales du code, les attributs, etc.

3.2 Rendu du projet

Le rendu du projet se fera au plus tard le **vendredi 28 avril à 18h** par l'un des membres du binôme dans le dépôt de son groupe de TP sur `madoc`. Aucun retard ne sera autorisé.

Vous devrez déposer une archive au format `zip` comportant un rapport au format `pdf` et les programmes sources. Pensez à faire un `make clean` avant de créer l'archive.

Le rapport sera constitué des 4 parties suivantes (et seulement de ces parties) :

1. Le couple SDA/SDC `Groupe` avec un petit texte résumant vos choix, les signatures des opérations en pseudo-code, la représentation mémoire choisie (illustrer graphiquement) et les complexités des opérations (faire un tableau) ;
2. Le couple SDA/SDC `Cortege` (comme pour les groupes) ;
3. Le couple SDA/SDC `Manif` (comme pour les groupes) ;
4. Des résultats de simulation avec les temps de calcul montrant la montée en charge avec des manifestations de plus en plus grosses. On peut au moins mettre en place les simulations suivantes :
   — simuler une manifestation du début à la fin (toutes les personnes ont fini de défiler) ;
   — simuler une manifestation de manière à ce que toutes les personnes défilent (tout le monde défile, personne n'est encore sorti) puis extraire toutes les personnes en retirant à chaque fois la dernière personne rentrée.

3.3 Absences aux TP

Le projet sera encadré pendant quatre séances de TP. Un appel sera fait à chaque séance. Il sera toléré au plus une absence non justifiée par personne. Si vous ne justifiez pas au moins deux absences, votre note finale sera 0.

3.4 Exigences et notation

Les exigences pour une SDC sont les suivantes :

— Au niveau A, la description de la représentation en mémoire est très claire et bien expliquée au moyen de graphiques. Les complexités des algorithmes sont optimales.
— Au niveau B, la description est celle du niveau A et les complexités sont bonnes sans être optimales.
— Au niveau C, la description n'est pas claire mais les complexités sont bonnes.
— Au niveau D, la description est absente ou incohérente ou les complexités sont mauvaises par rapport à ce qui est attendu.

Les exigences pour une SDA sont les suivantes :

— Au niveau A, le rôle de chaque opération ou méthode est bien expliqué. Les signatures sont définies et constituent une bonne interface pour un utilisateur. Les préconditions sont données. On peut utiliser la SDA facilement pour écrire de nouveaux algorithmes.
— Au niveau B, la SDA est globalement bonne mais les explications ne sont pas toujours limpides et l'interface peut être améliorée. La SDA peut quand même être utilisée pour écrire de nouveaux algorithmes.

— Au niveau C, il manque des préconditions, les rôles ne sont pas toujours expliqués clairement et l'interface peut être améliorée.

— Au niveau D, la SDA est inutilisable pour écrire de nouveaux algorithmes. Un utilisateur ne peut pas la comprendre.

Les exigences pour le code `C++` sont les suivantes :

— Au niveau A, le code est clair, bien commenté, bien indenté. Les classes standards de `C++` sont mises en œuvre à bon escient. La compilation ne génère pas de *warnings*. Les méthodes sont élémentaires, le code est bien découpé. Les résultats d'exécution sont corrects, i.e. les algorithmes sont corrects et bien codés.

— Au niveau B, le code est globalement d'un bon niveau mais il y a des faiblesses par rapport au niveau A. Par exemple, certaines méthodes ne sont pas élémentaires. Les résultats d'exécution sont corrects, i.e. les algorithmes sont corrects et bien codés.

— Au niveau C, le code compile mais il est mal commenté ou indenté (résultat d'un développement à la va-vite). Il n'y a pas de plantage à l'exécution mais certains résultats sont incorrects, i.e. certains algorithmes sont faux ou mal codés.

— Au niveau D, le code ne compile pas ou il plante à l'exécution. Le code n'est pas commenté. Il manque des parties du code.

Enfin, un barème est donné ci-dessous à titre indicatif :

— Entre 17 et 20 : niveau au moins B+ en moyenne, niveau A à B dans toutes les catégories ;

— Entre 13.5 et 16.5 : niveau B en moyenne, niveau au moins C+ dans toutes les catégories ;

— Entre 10 et 13 : niveau C+ en moyenne, solution en partie fonctionnelle ;

— Moins de 10 : niveau C ou D en moyenne, projet insuffisant comportant de nombreux manques.