Chaque classe devra être intégrée de manière modulaire, avec des fichiers .h et .cpp portant le nom de la classe. Fournir aussi le module du main (même s'il n'est pas noté) et le makefile.

Le but de ce sujet est de modéliser des parties du jeu de carte « La bataille » dans une version simplifiée. Le jeu se joue avec un jeu de 52 cartes classiques :

- 13 cartes par couleur : As, 2, …, 10, Valet, Dame, Roi
- 4 couleurs : trèfle, carreau, cœur, pique

Le jeu se joue à deux joueurs. Au début du jeu chaque joueur reçoit 26 cartes. A chaque tour du jeu, chaque joueur pose la première carte de son paquet. Le joueur ayant posé la carte la plus forte remporte les deux cartes et les met à la fin de son paquet. Le jeu se termine quand un joueur n'a plus de cartes. La carte la plus forte est l'As, puis le Roi, la Dame, le Valet, le 10, …, jusqu'au 2. A numéro identique, la couleur permet de départager, dans l'ordre pique, cœur, carreau, trèfle (du plus fort au moins fort).

On rappelle qu'un constructeur par défaut est un constructeur sans paramètre.

1) Le but est d'écrire une classe **Carte** pour représenter une carte d'un jeu classique de 52 cartes.

```
class Carte
{
…
private :
    int numero;
    int couleur;
} ;
```

Le numéro prend sa valeur dans {2,…, 13, 14}, avec 11, 12, 13, 14 représentant respectivement le Valet, la Dame, le Roi et l'As.
La couleur prend sa valeur dans {1,..,4}, représentant respectivement trèfle, carreau, cœur et pique.
Ecrire :

a) Un constructeur prenant en arguments un numéro et une couleur et un constructeur par défaut qui initialise numéro et couleur à 0.
b) Un opérateur << d'affichage, qui fait des affichages du style As de Pique, 7 de Trefle, Dame de Carreau, etc. Pour la carte construite avec le constructeur par défaut on affichera "Carte nulle".
c) Des fonctions *getNumero*, *getCouleur*, *setNumero* et *setCouleur* permettant de récupérer le numéro ou la couleur, ou de les modifier.
d) Des opérateurs >, <, == et != de comparaison de deux cartes.

2) Il s'agit d'écrire une classe **ListeCartes** pour représenter une suite de cartes, dans laquelle on peut trouver des cartes identiques (par ex. deux 7 de Trèfle).
Les cartes sont stockées dans un tableau alloué dynamiquement. La liste est créée vide, des fonctions permettent ensuite d'ajouter ou retirer un carte.

```
class ListeCartes
{
…
private :
    Carte *cartes;          tableau de cartes, alloué par new
    int taille;             taille du tableau de cartes, c'est le nombre max. de cartes qu'on
                            peut ajouter à la liste
    int nbCartes;           nombre de cartes ajoutées à la liste (varie de 0 à taille)
};
```

Ecrire :

a) Un constructeur qui reçoit la taille et initialise la liste à vide (nb de cartes à 0).
b) Un destructeur, un constructeur par copie et un opérateur =.
c) Une fonction *void ajouter(const Carte& ca)* d'ajout d'une carte en fin de liste et une fonction *Carte extraire()* qui extrait la première carte (la retire de la liste et la retourne, la suppression nécessitant le décalage des cartes qui suivent).
d) Une fonction d'affichage de la liste.
e) Une fonction qui teste si la liste contient une carte donnée (reçue en paramètre).
f) Une fonction qui retourne le nombre de cartes présentes dans la liste.
g) Une fonction qui mélange les cartes. Pour cela, on pourra répéter 1000 fois : sélectionner deux cartes au hasard et les inverser. Voir en fin de document les indications sur les fonctions de tirage aléatoire.

3) L'objectif est d'écrire une classe **Paquet** pour représenter un paquet de cartes. Elle hérite de ListeCartes : un paquet est une liste de cartes qui contient au maximum 52 cartes et où on ne trouve pas deux cartes identiques.
Elle n'ajoute pas de donnée membre par rapport à ListeCartes.
Ecrire :

a) Un constructeur sans paramètre initialisant à un paquet vide.
b) La redéfinition de la fonction *ajouter* de ListeCartes (même nom et prototype) afin de ne pas faire l'ajout si une carte identique est déjà présente dans le paquet.
c) Une fonction statique *Paquet creerPaquetComplet()* qui crée et retourne un paquet de 52 cartes avec toutes les cartes du jeu.

4) Nous introduisons maintenant une classe **Joueur**. Elle a une unique donnée membre privée, de type Paquet, qui représente le paquet de cartes du joueur.
Ecrire :
a) Un constructeur par défaut qui crée un joueur possédant un paquet vide.
b) Une fonction *void prendreCarte(const Carte& ca)* qui ajoute la carte en fin du paquet du joueur.
c) Une fonction *Carte jouerCarte()* qui retire du paquet du joueur la première carte et la retourne.
d) Une fonction d'affichage des cartes du joueur.
e) Une fonction qui retourne le nombre de cartes du joueur.

5) Nous créons enfin une classe **Partie**. Elle modélise une partie simplifiée du jeu de cartes « La bataille », dont les règles sont rappelées en début de sujet.

```
class Partie
{
…
private :
    Joueur joueur1, joueur2;
    bool termine;
} ;
```

Ecrire :
a) Un constructeur par défaut. Il distribue aléatoirement 26 cartes à chaque joueur (le caractère aléatoire sera obtenu en générant un paquet de 52 cartes et en le mélangeant).
b) Une fonction *int jouer()* qui joue la partie : tant qu'il reste des cartes aux deux joueurs, chaque joueur pose une carte, le vainqueur du pli ramasse les deux cartes. Elle retourne le numéro du joueur victorieux.
Attention : un objet Partie ne sert à qu'à faire une seule partie, le booléen *termine* permet à la fonction *jouer* de refuser une partie supplémentaire.

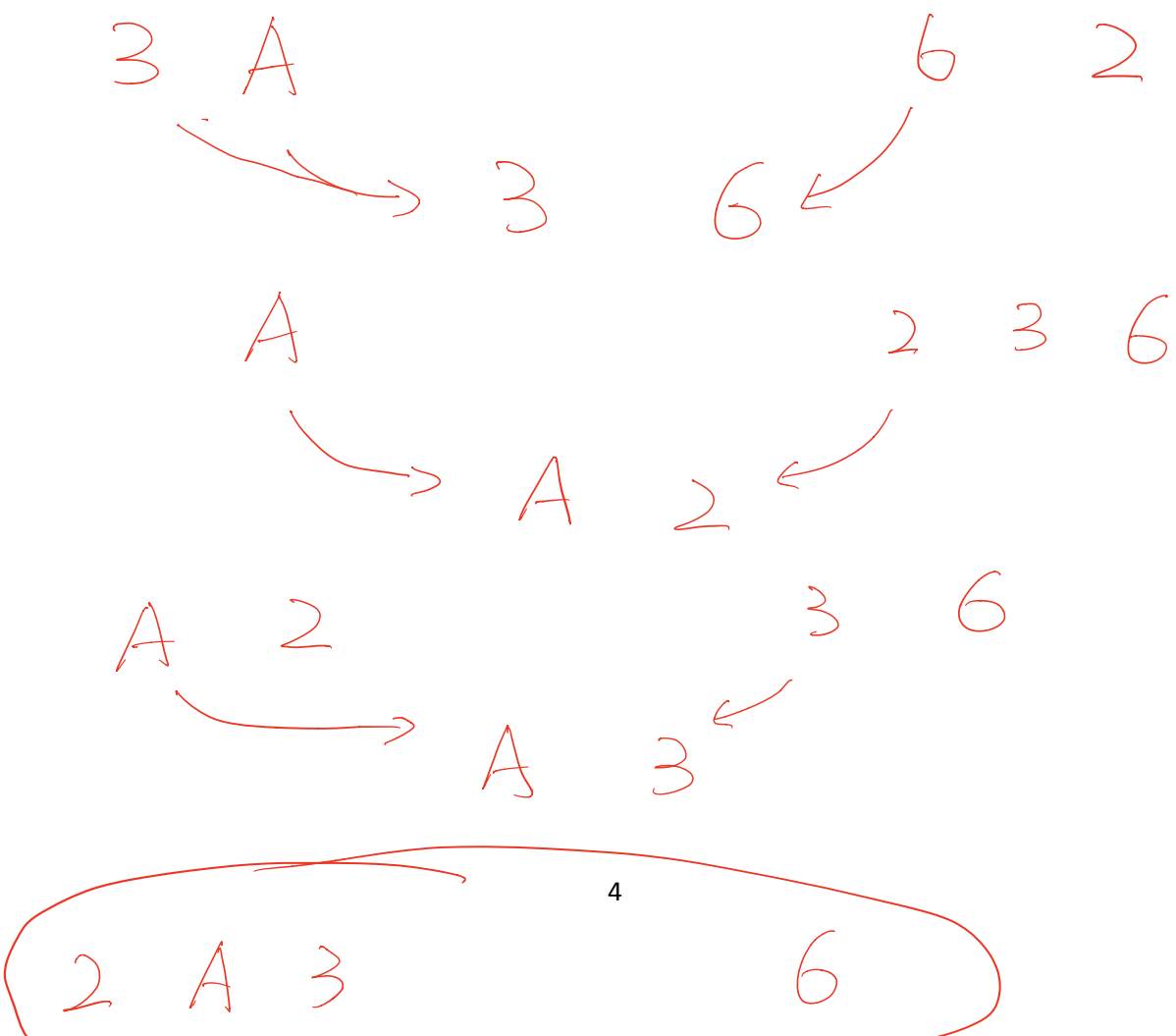
6) Nous souhaitons pouvoir **tracer** (afficher) le déroulement détaillé des parties : distribution initiale, puis à chaque pli, les cartes du pli et le nombre de cartes de chaque joueur une fois le pli ramassé.
Ajouter une donnée membre booléenne statique indiquant si on est en mode trace ou pas et une fonction statique permettant d'activer ou désactiver ce mode. Modifier la fonction *jouer()* pour faire les affichages souhaités lorsque le mode trace est actif.

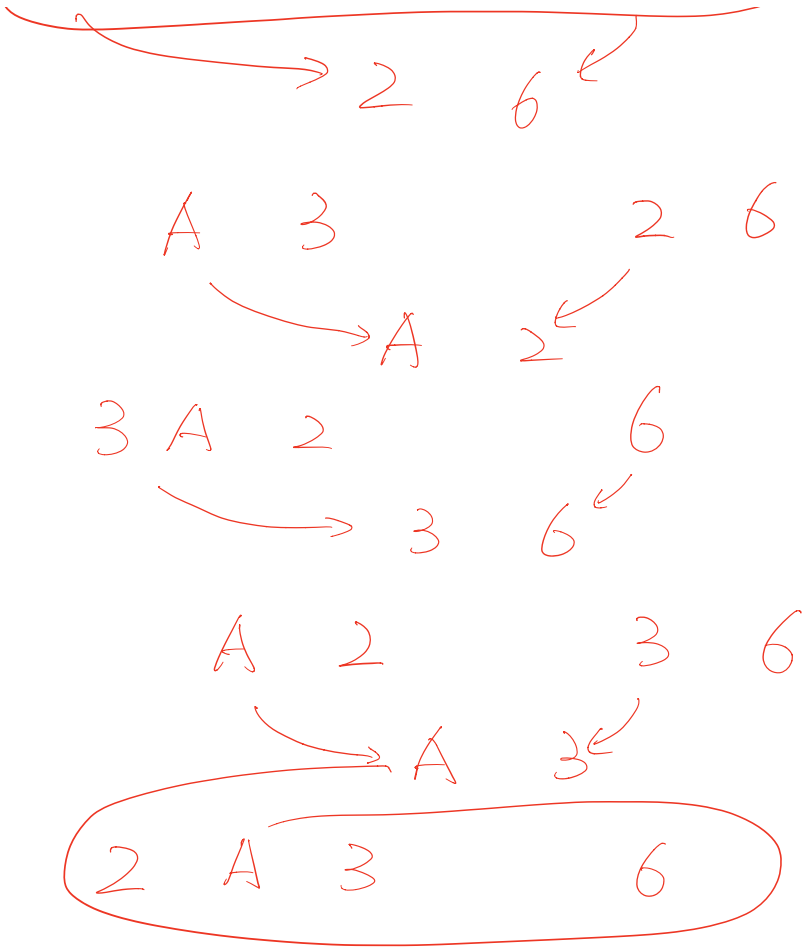
7) Nous souhaitons finalement faire quelques **statistiques** sur les parties : pourcentage de victoires de chaque joueur et nombre de coups moyen par partie.
   a) Ajouter des données membre à la classe Partie permettant d'obtenir ces statistiques.
   b) Ajouter une fonction statique d'affichage des statistiques.

Indications de programmation :

Pour faire des tirages aléatoires :

1) A inclure : <stdlib.h> et <time.h>
2) *srand(time(NULL));* // pour amorcer le mécanisme; à placer au début du main
3) des appels successifs à *rand()* retournent des nombres entiers aléatoires




2 6
A 3 2 6
A 2
3 A 2 6
3 6
A 2 3 6
A 3
2 A 3 6