TP 10 : Le jeu des prisonniers

Le dilemme des prisonniers est la situation suivante. Deux complices sont arrêtés, ils sont interrogés séparément et ne peuvent pas communiquer. On propose à chacun un marché. Soit il dénonce son complice, soit il refuse de parler. Si chacun des deux dénonce l'autre, ils ont chacun une petite remise de peine de un an. Si tous deux refusent de parler, ils ont chacun une remise de peine de 3 ans faute de preuves. Si l'un dénonce son complice qui refuse de parler, il bénéficie d'une remise de peine de 5 ans mais le complice reçoit la peine maximale.

On modélise cette situation dans un jeu où les deux joueurs choisissent simultanément et sans concertation de *Trahir* (dénoncer dans la situation précédente) ou de *Coopérer* (se taire). S'ils trahissent tous les deux, ils gagnent chacun un point. S'ils coopèrent tous les deux, ils gagnent chacun 3 points. Sinon, celui qui trahit gagne 5 points et celui qui coopère ne gagne aucun point.

Le jeu qui nous intéresse précisément est le dilemme des prisonniers *itéré* : les joueurs vont jouer une série de coups en connaissant à chaque fois uniquement ce qu'a joué l'autre aux coups précédents, et on cumule les scores obtenus. Il s'agit de définir une stratégie pour ce jeu et de l'implémenter en Maple, après quoi toutes les stratégies joueront les unes contres les autres, et seront classées par nombre de points obtenus.

1. Écrivez une procédure `Points` qui prend en argument deux coups `c1` et `c2` qui peuvent chacun valoir `'C'` (coopérer) ou `'T'` (trahir), et qui renvoie une liste à deux éléments : le nombre de points obtenus par chacun des joueurs.

   Par exemple, `Points('C','T')` doit renvoyer `[0,5]`.

   Remarque : il est important de respecter la convention sur le nom des coups possibles pour que votre stratégie soit compatible avec les autres et puisse participer au tournoi.
2. Écrivez une procédure `PointsCumules` qui prend en argument une liste de listes de deux coups, et qui renvoie une liste à deux éléments : le nombre de points totaux obtenus par chacun des joueurs.

   Par exemple, `PointsCumules([['C','C'],['C','T'],['T','T']])` renvoie `[4,9]`.

Une stratégie sera représentée en Maple par une procédure qui prend en argument une liste `L` de listes de deux coups : la première composante est le coup joué par le joueur lui-même, la seconde est le coup joué par son adversaire (par exemple, `L[1][2]` est le coup joué par mon adversaire au premier tour ; `L[nops(L)-1][1]` est le coup que j'ai joué au tour précédent), et qui renvoie `'C'` ou `'T'` pour signifier le prochain coup joué par la stratégie.

3. La stratégie *Gentil* consiste à toujours coopérer, la stratégie *Méchant* consiste à toujours trahir. Écrivez deux procédures `Gentil` et `Mechant` représentant ces deux stratégies.

4. La stratégie *Aléatoire* consiste à tirer au hasard si on coopère ou si on trahit. Écrivez une procédure `Aleatoire` représentant cette stratégie.

   Remarque : on peut obtenir un nombre entier aléatoire en Maple avec `rand()` ; par exemple si on veut un nombre aléatoire entre 0 et 2, on appelle `rand() mod 3`.

5. La stratégie *Rancunier* consiste à coopérer si l'autre n'a jamais trahi, et à trahir si l'autre a trahi au moins une fois. Écrivez une procédure `Rancunier` représentant cette stratégie.

6. La stratégie *Agressif* consiste à compter le nombre de fois que les joueurs ont joué un coup différent, et le nombre de fois que les joueurs ont joué le même coup. Si le premier compte est supérieur ou égal au second, trahir, sinon, coopérer. Écrivez une procédure `Agressif` représentant cette stratégie.

7. Inventez une stratégie et écrivez une procédure la représentant !

   Remarque : si vous avez besoin de savoir à quel tour on en est, il suffit de regarder la taille de la liste `L`.