Tour de cartes

Deux joueurs vont montrer un tour de cartes avec un jeu de 52 cartes. Pour simplifier, les cartes ont des valeurs distinctes entre 0 et 51.
Les cartes sont placées en ligne face visible sur une table dans un ordre inconnu des joueurs.
Le premier joueur va à la table, regarde les cartes et fait au plus S échanges. Chaque échange s'effectue en choisissant deux cartes aux positions i et j (i et j peuvent être égaux) et en déplaçant la carte de la position i à la position j et vice versa.
Ensuite, le premier joueur part sans communiquer avec le deuxième joueur et toutes les cartes sont retournées (faces cachées) sans en changer l'ordre. Le deuxième joueur est alors invité à la table et on lui demande de deviner où se trouve la carte dont la valeur est **cible** ; il peut retourner jusqu'à T cartes une par une. Si l'une des cartes révélées a la valeur **cible**, alors les joueurs gagnent. S'il a épuisé ses tentatives, ils perdent.
Votre objectif est d'écrire deux programmes qui simulent l'action des joueurs et leur permet de gagner le jeu.

Détails d'implémentation

On vous fournira deux programmes, FirstPlayer et SecondPlayer, ainsi qu'un évaluateur d'exemple.

Dans FirstPlayer, vous devez implémenter la fonction suivante :

```
void swapCards(int cards[], int S, int T)
```

- Cette fonction est appelée exactement une fois par l'évaluateur.
- Cards : un tableau contenant la valeur initiale des cartes de gauche à droite, avec exactement 52 éléments indexés de 0 à 51
- S : le nombre d'échanges autorisés
- T : le nombre de tentatives autorisées

`swapCards` peut appeler la fonction suivante :

```
void doSwap(int i, int j)
```

- i : indice de la première carte à échanger, $0 \le i < 52$
- j : indice de la deuxième carte à échanger, $0 \le j < 52$
- `doSwap` peut être appelée au plus S fois

Dans SecondPlayer, vous devez implémenter la fonction suivante :

```
void guessCard(int S, int T, int target)
```

- S : le nombre d'échanges autorisés
- T : le nombre de tentatives autorisées
- target : la valeur de carte qu'il faut révéler (la cible)

`guessCard` peut appeler la fonction suivante :

```
int guess(int idx)
```

- idx : la position de la carte à révéler, $0 \leq idx < 52$
- Elle retourne la valeur de la idx-ème carte
- `guess` peut être appelée au plus T fois
- lorsqu'une tentative révèle la bonne carte, l'évaluateur arrête le programme.

Exemple d'interaction

Voici un exemple d'entrée pour l'évaluateur et les appels de fonctions correspondant.
La première ligne doit contenir deux entiers : S et T.
La deuxième ligne doit contenir 52 nombres : le ième nombre dénote la valeur de la ième carte.
La troisième ligne contient l'entier **cible.**

Exemple d'entrée pour l'évaluateur	Exemples d'appels		
	Appels	Sous-appels	Renvois
1 51 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 1	swapCards([0,1,…], 1, 51)		
		doSwap(0, 1)	
			échange les cartes aux positions 0 et 1
	swapCards termine		.
	guessCard(1, 51, 1)		
		guess(5)	
			guess renvoie 5

		guess(1)	
			guess renvoie 0
		guess(0)	
			Correct !

Contraintes

- $1 \leq S \leq 52$
- $1 \leq T \leq 51$
- $0 \leq cible < 52$

Sous-tâches

1. (16 points) : $S = 52,\ T = 1$
2. (20 points) : $S + T = 52$
3. (22 points) : $S = 13,\ T = 27$
4. (18 points) : $S = 1,\ T = 26$
5. (24 points) : Il existe une stratégie gagnante pour les S et T donnés