1/LOTO

On veut analyser les résultats de différentes grilles de loto.
On dispose des types suivants :

```
typedef int GRILLE[49] ;
typedef int TIRAGE[7];
```

On suppose qu'une grille de loto est stockée dans un tableau G de type GRILLE, sachant que G[i] = 1 si et seulement si le numéro i+1 a été coché par le joueur.
On dispose d'autre part d'un tableau T de type TIRAGE dont les six premières cases contiennent les six numéros sortis lors du tirage et la septième case contient le numéro complémentaire.

1. Ecrire une fonction *présent* ayant en paramètre une grille G et un numéro N et qui renvoie 1 si et seulement si le numéro N a été coché dans la grille G.
2. Ecrire une fonction *numéros* qui prend comme paramètre une grille G et un tirage T et qui renvoie le nombre de numéros parmi les six premiers du tirage T qui ont été cochés dans la grille G.
3. Ecrire un sous programme *résultat* qui affiche pour une grille G donnée et un tirage T, le résultat en fonction des catégories : 6 bons numéros, 5 bons numéros + complémentaire, 5 bons numéros, 4 bons numéros + complémentaire, 4 bons numéros, 3 bons numéros + complémentaire, 3 bons numéros ou rien.

2/Mystère

Donner la suite des instructions exécutées, les appels récursifs et les valeurs retournées dans le cas de l'appel du sous-programme suivant avec pour paramètre la valeur 4. Que calcule ce sous-programme ?

```
int mystere (int n)
{
        if (n == 1)
                return 1;
        else
                return mystere(n-1) + 2*(n-1) + 1;
}
```

3/MASTER-MIND

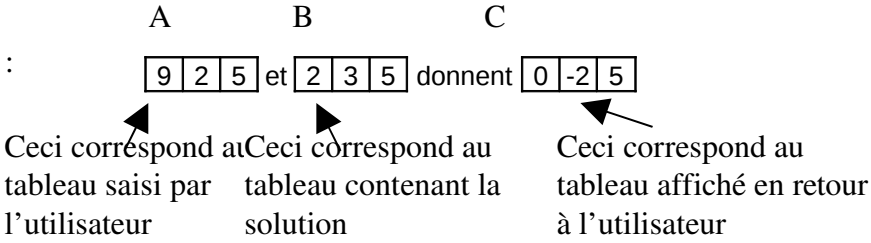
```
#include<stdio.h>
typedef ...TAB... ;
TAB A,B,C;
int gagne;

int main()
{
gagne = 1; // initialisé à faux
creer-tableau-aleatoire(B) ; /*crée un tableau contenant des valeurs aléatoires différentes comprises entre 1 et 9*/
do
        saisie-tab(...) ;
```

```
        compare-tab(...) ;
        affiche-tab(...) ; //affiche le tableau C
while(gagne == 1) ;
printf(' Bravo, vous avez gagné' );
}
```

On suppose le sous-programme creer-tableau-aleatoire déjà connue.

1) Définir le type TAB qui est un tableau d'entiers de 3 cases.
2) Ecrire le sous-programme **saisie-tab** qui permet à l'utilisateur de remplir un tableau de type TAB avec des nombres compris entre 1 et 9.
3) Ecrire le sous-programme **affiche-tab** qui affiche un tableau de type TAB.
4) Ecrire le sous-programme **appartient** qui teste si un entier X appartient ou non à un tableau B de type TAB (on retournera 0 pour vrai ou 1 pour faux).
5) Ecrire le sous-programme **compare-tab** qui compare deux tableaux A et B de type TAB et remplit un troisième tableau C de la manière suivante (vous pourrez utiliser les programmes précédents) :
   - Si une case **x** des tableaux A et B possède la même valeur **v**, alors C prend la valeur **v**
   - Si la valeur **v** d'une case **x** de A apparaît dans une case de B alors la case **x** de C prend la valeur **-v**
   - Si la valeur **v** d'une case **x** de A n'apparaît dans aucune case de B alors la case **x** de C prend la valeur 0

   Exemple :

   Si toutes les cases de A sont identiques à B alors la variable **gagne** prend la valeur 0 sinon elle prend la valeur 1.
6) Que faudrait-il ajouter au programme principal pour limiter le nombre d'essais à 5.

Question subsidiaire : Ecrire le sous-programme **creer-tableau-aleatoire.**

4/Puissance

1) Écrire une fonction " puissance " qui calcule X^n pour X réel et n entier positif ou négatif quelconque.

2) On peut remarquer que lorsque n est positif : $X^n = \begin{cases} X.X^{n-1} \text{ si n est impair} \\ (X^2)^{\frac{n}{2}} \text{ si n est pair} \end{cases}$

Ecrire une fonction "puissanceV2 " qui calcule X^n, en tenant compte de la remarque précédente. Montrer que cette seconde version est plus rapide, en présentant les emboîtements effectués par les deux fonctions avec une valeur de n pas trop petite.