Énoncé de l'exercice

$$Z = a + ib \quad \text{avec } (a, b) \in \mathbb{R}^2$$

- Constituer une file de nombres complexes.
- La file n'excédera pas 100 éléments.
- Quand on défile un élément, on retourne sa norme ($|Z| = \sqrt{a^2 + b^2}$)
- Implémentez les fonctions suivantes
    - Initialisation
    - Taille de la file
    - Vérifier si la file est vide ou pleine
    - Enfiler
    - Défiler
    - Calculer le nombre d'occurrence d'un élément dans la file

---

Fichier en-tête

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define VIDE 0
#define PLEIN 100

typedef struct Complexe{
        double a;
        double b;
}Complexe;

typedef struct Noeud{
        Complexe z;
        struct Noeud *next;
}Noeud;

typedef struct File {
        int nbrElement;
        Noeud *debut;
        Noeud *fin;
}File;

typedef enum boolean{faux, vrai} boolean;

void initFile(File *);
int size(File *);
boolean empty(File *);
boolean full(File *);
boolean enfiler(Complexe, File*);
double defiler(File *);
int count(Complexe, File *);
```

Les fonctions

Initialisation

```
void initFile(File *maFile){
        maFile -> nbrElement = 0;
        maFile -> debut = NULL;
        maFile -> fin = NULL;
}
```

Taille de la file

```
int size(File *maFile){
        return maFile -> nbrElement;
}
```

Vérifier si la file est vide

```
boolean empty(File *maFile){
        return (maFile -> nbrElement == VIDE);
}
```

Vérifier si la file est pleine

```
boolean full(File *maFile){
        return (maFile -> nbrElement == PLEIN);
}
```

Enfiler

```
boolean enfiler(Complexe nombre, File *maFile){

        // si la file n'est pas pleine, alors on peut ajouter le Noeud
        if (!full(maFile))
        {
                // Création du Noeud
                Noeud *ajout = (Noeud *)malloc(sizeof(Noeud));
                ajout -> z = nombre;
                ajout -> next = NULL;

                if (empty(maFile))
                {
                        maFile -> debut = ajout;
                        maFile -> fin = ajout;
                        maFile -> nbrElement++;
                        return vrai;
                }

                maFile -> fin -> next = ajout;
                maFile -> fin = ajout;
                maFile -> nbrElement++;
                return vrai;
        }

        // On retourne faux si la liste est pleine
        return faux;
}
```

Défiler

```
double defiler(File *maFile){
        double result = 0.0;

        if (!empty(maFile))
        {
                Noeud *tmp = maFile -> debut;
                Complexe z = tmp -> z;
                result = sqrt((z.a * z.a) + (z.b * z.b));

                maFile -> debut = tmp -> next;
                free(tmp);
                maFile -> nbrElement--;
        }

        return result;
}
```

Calculer le nombre d'occurrence d'un élément dans la liste

```
int count(Complexe cherche, File *maFile){
        int occ = 0;

        Noeud *tmp = maFile -> debut;
        int i;
        for(i = 0; i < size(maFile); i++)
        {
                if ((tmp->z.a == cherche.a) && (tmp->z.b == cherche.b))
                        occ++;
                tmp = tmp -> next;
        }

        return occ;
}
```

Fonction principale

Vous êtes libres d'écrire la fonction principale comme bon vous semble.

```
int main(){
    File maFile;

    initFile(&maFile);

    empty(&maFile) ? printf("Vide\n") : printf("Non vide\n");

    printf("Taille = %d \n", size(&maFile));

    Complexe c1, c2, c3;

    c1.a = 1;
    c1.b = 2;

    c2.a = 3;
    c2.b = 4;

    c2.a = 5;
    c2.b = 6;

    (enfiler(c1, &maFile)) ? printf("c1 ajoute\n") : printf("echec");
```

```
    (enfiler(c2, &maFile)) ? printf("c2 ajoute\n") : printf("echec");

    (enfiler(c1, &maFile)) ? printf("c1 ajoute\n") : printf("echec");

    (enfiler(c3, &maFile)) ? printf("c3 ajoute\n") : printf("echec");

    (enfiler(c1, &maFile)) ? printf("c1 ajoute\n") : printf("echec");


    printf("La taille actuelle de la file est %d\n", size(&maFile));

    printf("c1 apparait %d fois\n", count(c1, &maFile));

    printf("J'ai enleve c1 sa norme est %f \n", defiler(&maFile));

    printf("La nouvelle taille est %d\n", size(&maFile));

    return 0;
}
```

> **Cette fonction principale, affiche le texte suivant :**
>
> Vide
> Taille = 0
> c1 ajoute
> c2 ajoute
> c1 ajoute
> c3 ajoute
> c1 ajoute
> La taille actuelle de la file est 5
> c1 apparaît 3 fois
> J'ai enlevé c1 sa norme est 2.236068
> La nouvelle taille est 4