Nous souhaitons utiliser une liste linéaire simplement chaînée pour l'élection présidentielle de 2022, c'est-à-dire la liste des électeurs.

Pour chaque électeur, on va stocker son nom, son numéro de carte d'identité ainsi que son choix de vote.

A. Structures de données et types :

Définir la structure **electeur** qui comporte les champs suivants :

- Le **nom** de type *chaine de caractères*,
- Le **cin_num** de type *long*,
- Le **choix** de type *entier*
- Un pointeur sur l'électeur **suivant.**

N.B.: Pour limiter le calcul, nous définissons le choix comme suit ; NOM1 : 1 ; NOM2 : 2 ; NOM3 : 3 ; NOM4 : 4 ; et tout autre choix : 5.

Définir le type **T_Electeur** comme un pointeur sur la structure ***struct electeur***.

B. Fonctions à implémenter :

1. Ecrire une fonction qui permet de créer un électeur. Le prototype est :

   T_Electeur creationelecteur(void);

2. Ecrire une fonction qui affiche la liste des électeurs.

   void afficheliste(T_Electeur);

3. Le nombre d'électeurs s'incrémente tout au long de la journée et selon les centres électoraux. Ecrire une fonction qui permet d'ajouter un électeur par ordre alphabétique de son nom dans la liste:

   void ajoutelecteur(T_Electeur *, char [], long, int);

4. Ecrire une fonction qui renvoie le nombre des électeurs de la liste. Le prototype de la fonction est comme suit :

   int comptelecteur(T_Electeur);

5. Ecrire une fonction qui recherche un électeur avec le numéro de sa carte d'identité et affiche toutes les données le concernant. Le prototype de cette fonction est :

   int trouvelecteur(T_Electeur, long);

6. Sachant qu'il existe des électeurs ajoutés qui ne satisfaites pas les conditions d'élection. Il faut donc les supprimer de la liste. Ecrire la fonction qui permet de supprimer un électeur à l'aide de son numéro CIN de n'importe quelle position de la liste. Le prototype de cette fonction de suppression est la suivante :

   void Supprimelecteur(T_Electeur *,long);

7. Ecrire une fonction qui décompose la liste des électeurs en trois sous-listes **gauche**, **droite** et **blanc** selon le choix de candidats. Nous considérons que les électeurs sont :

   a. Dans la liste **gauche** s'ils ont choisi NOM1 ou NOM3 ;

   b. Dans la liste **droite** s'ils ont choisi NOM2 ou NOM4 ;

c. Et dans la liste **blanc** s'ils ont fait les autres choix.

Le prototype de cette fonction est le suivant :

void decoupeliste(T_Electeur,T_Electeur *,T_Electeur *,T_Electeur*);

8. Après avoir séparé les trois listes, nous allons les trier selon le numéro de la carte d'identité. Le prototype de la fonction de tri de liste est comme suit :
   void triliste(T_Electeur);
9. Ecrire une fonction qui crée une nouvelle liste en fusionnant seulement les deux sous-listes de **gauche** et de **droite**. Cette fonction de fusion doit conserver l'ordre des électeurs selon le numéro de la carte d'identité. Le prototype de cette fonction est :

   T_Electeur fusionlistes(T_Electeur, T_Electeur);
10. Pour faire une estimation pour le second tour, nous calculons le *pourcentage* dans la liste fusionnée des électeurs *de **gauche** et de **droite***. Pour réaliser ce calcul, écrire une fonction qui rend le nombre des électeurs de **gauche** dans la liste fusionnée. Le prototype de la fonction est :

    int compteGD(T_Electeur);
11. Ecrire une fonction pour libérer une liste.
    void libereliste(T_Electeur);

C. Interface :

Ecrire les fonctions précédentes pour écrire un programme permettant de gérer l'élection. Le programme propose un menu qui contient les fonctionnalités suivantes :

1. Entrer les électeurs,
2. Ajouter des électeurs,
3. Supprimer le dernier électeur,
4. Rechercher un électeur,
5. Afficher la liste des électeurs,
6. Calculer le nombre des électeurs,
7. Découper la liste en trois sous-listes selon les choix: droite, gauche et blanc
   a. Trier les sous-listes
   b. Afficher les sous-listes,
   c. Fusionner les deux sous-listes : gauche et droite
8. Calculer les pourcentages de gauche et de droite pour le 2ème tour,
9. Libérer les listes
10. Quitter

D. Consignes générales:

À la fin du programme, les blocs de mémoire dynamiquement alloués doivent être proprement libérés.

L'organisation MINIMALE du projet est la suivante :

- Fichier d'en-tête **tp3.h**, contenant la déclaration des structures/fonctions de base,
- Fichier source **tp3.c**, contenant la définition de chaque fonction,

- Fichier source **main.c**, contenant le programme principal.

Votre rapport de quatre pages maximum contiendra la liste des structures et des fonctions supplémentaires que vous avez choisi d'implémenter et les raisons de ces choix.

Un exposé succinct de la complexité de chacune des fonctions implémentées.

Votre rapport et vos trois fichiers feront l'objet d'une remise de devoir sur **Moodle** dans l'espace qui sera ouvert à cet effet (un seul rendu de devoir par binôme).