```
/*
 ------------------------------------------------------------------------------
 Nom du fichier : main.h
 Auteur(s)      : Romain Fleury, Nicolet Victor
 Date creation  : 26.04.2023

 Description    : Programme de test pour la librairie "listes_dynamique"

 Remarque(s)    : -

 Compilateur    : Mingw-w64 gcc 12.2.0
 ------------------------------------------------------------------------------
*/
#include <stdio.h>
#include "listes_dynamiques.h"

bool critere(size_t pos, const Info *n) {
    //return true;
    if (pos % 2 == 0 || *n > 1 && *n < 6) {
        return true;
    }
    return false;
}

int main() {
    size_t position = 0;
    int tailleListe;
    Info x = 0;

    printf("-----====### INITIALISATION ###==========-----\n");
    Liste *lptr1 = initialiser();

    printf("\nINITIALISATION : tableau de variable et de pointeur.\n");
    printf("%-20s %-20s %-20s \n", "Variable", "Adresse", "Adresse pointeur");
    printf("%-20s %-20p %-20p \n", "lptr", (void *) &lptr1, (void *) lptr1);
    printf("%-20s %-20p %-20p \n", "lptr->tete", (void *) &(lptr1->tete),
           lptr1->tete);
    printf("%-20s %-20p %-20p \n", "lptr->queue", (void *) &(lptr1->queue),
           lptr1->queue);

    printf("\nINITIALISATION : test des fonctions estVide() et longueur().\n");
    printf("%-20s : %d \n", "Liste est vide", estVide(lptr1));
    printf("%-20s : %zu \n", "Taille de la liste", longueur(lptr1));

    printf("\nINITIALISATION : Affichage d'une liste vide.\n");
    printf("FORWARD -> ");
    afficher(lptr1, FORWARD);
    printf(" et BACKWARD -> ");
    afficher(lptr1, BACKWARD);
    printf("\n");

    printf("\n----------### SUPPRIMER ET INSERER EN TETE ###---------\n");

    printf("\nSUPPRIMER EN TETE : utiliser supprimerEnTete() sur une liste vide.\n");
    printf("%-18s : %d \n%-18s : ", "-Status",
           supprimerEnTete(lptr1, &x), "-Liste");
    afficher(lptr1, FORWARD);
    printf("\n");

    printf("\nINSERER EN TETE : utiliser insererEnTete() pour "
           "inserer les valeurs de 0 a 5.\n");
    x = 0;
    printf("%-18s : %d \n%-18s : ", "-Status",
           insererEnTete(lptr1, &x), "-Liste");
    for (Info i = 1; i <= 5; ++i)
        insererEnTete(lptr1, &i);
    afficher(lptr1, FORWARD);
    printf("\n");

    printf("\nSUPPRIMER EN TETE : utiliser supprimerEnTete() sur une liste.\n");
    printf("%-18s : %d \n", "-Status", supprimerEnTete(lptr1, &x));
    printf("%-18s : %d \n%-18s : ", "-Valeur supprimee", x, "-Liste");
    afficher(lptr1, FORWARD);
    printf("\n");
```

```
    printf("\n----------### SUPPRIMER ET INSERER EN QUEUE ###=========\n");

    Liste *lptr2 = initialiser();
    printf(
        "\nSUPPRIMER EN QUEUE : utiliser supprimerEnQueue() sur une liste vide.\n");
    printf("%-18s : %d \n%-18s : ", "-Status",
             supprimerEnQueue(lptr2, &x), "-Liste");
    afficher(lptr2, FORWARD);
    printf("\n");

    printf("\nINSERER EN QUEUE : utiliser insererEnQueue() pour "
             "inserer les valeurs de 0 a 5.\n");
    x = 0;
    printf("%-18s : %d \n%-18s : ", "-Status",
             insererEnQueue(lptr2, &x), "-Liste");
    for (Info i = 1; i <= 5; ++i)
        insererEnQueue(lptr2, &i);
    afficher(lptr2, FORWARD);
    printf("\n");

    printf("\nSUPPRIMER EN QUEUE : utiliser supprimerEnQueue() sur une liste.\n");
    printf("%-18s : %d \n", "-Status", supprimerEnQueue(lptr2, &x));
    printf("%-18s : %d \n%-18s : ", "-Valeur supprimee", x, "-Liste");
    afficher(lptr2, FORWARD);
    printf("\n");

    printf("\n----------### VIDER ###====-----");

    printf("\n\nVIDER : utiliser vider() sur une liste vide.\n");
    Liste *listeTestVider = initialiser();
    printf("%-45s : ", "-Initilisation d'une liste vide");
    afficher(listeTestVider, FORWARD);
    vider(listeTestVider, position);
    printf("\n%s %-3zu : ", "-Apres utilisation de vider() en position", position);
    afficher(listeTestVider, FORWARD);

    printf("\n\nVIDER : utiliser vider() sur une liste de un element.\n");
    printf("%-45s : ", "-Ajout d'un element dans la liste");

    x = 0;
    insererEnQueue(listeTestVider, &x);
    afficher(listeTestVider, FORWARD);
    printf("\n%s %-3zu : ", "-Apres utilisation de vider() en position", position);
    vider(listeTestVider, position);
    afficher(listeTestVider, FORWARD);

    printf("\n\nVIDER : utiliser vider() sur une liste de plusieurs elements.\n");
    printf("%-45s : ", "-Ajout de plusieurs elements dans la liste");

    tailleListe = 10;
    for (Info i = 0; i < tailleListe; ++i)
        insererEnQueue(listeTestVider, &i);
    afficher(listeTestVider, FORWARD);

    position = 5;
    printf("\n%s %-3zu : ", "-Apres utilisation de vider() en position", position);
    vider(listeTestVider, position);
    afficher(listeTestVider, FORWARD);

    position = 2;
    printf("\n%s %-3zu : ", "-Apres utilisation de vider() en position", position);
    vider(listeTestVider, position);
    afficher(listeTestVider, FORWARD);

    position = 0;
    printf("\n%s %-3zu : ", "-Apres utilisation de vider() en position", position);
    vider(listeTestVider, position);
    afficher(listeTestVider, FORWARD);
```

```
    printf(
        "\n\nVIDER : utiliser une position plus grande que la taille de la liste\n");

    tailleListe = 5;
    position = 6;
    for (Info i = 0; i < tailleListe; ++i)
        insererEnQueue(listeTestVider, &i);

    printf("%-59s : ", "-Liste de base");
    afficher(listeTestVider, FORWARD);
    printf("\n%s %zu %s %d : ", "-Utiliser vider() en position",
             position, "sur une liste de longueur", tailleListe);
    vider(listeTestVider, position);
    afficher(listeTestVider, FORWARD);
    printf("\n%-59s : ", "-Vider completement la liste");

    position = 0;
    vider(listeTestVider, position);
    afficher(listeTestVider, FORWARD);

    position = 3;
    printf("\n%s %zu %-27s : ", "-Utiliser vider() en position",
             position, "sur une liste vide");
    vider(listeTestVider, position);
    afficher(listeTestVider, FORWARD);
    printf("\n");

    printf("\n----------### ELEMENTS ###====-----\n");

    tailleListe = 5;
    printf("\nELEMENT : initialisation et ajout de %d valeur dans une liste.\n",
             tailleListe);
    Liste *lptr3 = initialiser();
    for (Info i = 0; i < tailleListe; ++i)
        insererEnTete(lptr3, &i);

    printf("%s : ", "-Liste apres initialisation et ajouts d'elements");
    afficher(lptr3, FORWARD);
    printf("\n");
    printf("%-48s : %d \n",
             "-Element de tete", lptr3->tete->info);
    printf("%-48s : %d \n",
             "-Element suivant de tete", lptr3->tete->suivant->info);
    printf("%-48s : %d \n",
             "-Element de queue", lptr3->queue->info);
    printf("%-48s : %d \n",
             "-Element precedent de queue", lptr3->queue->precedent->info);
    printf("\n");

    printf("----------### EGALITE ###====-----\n");

    printf("\nEGALITE : 1 = egales / 0 = non-egales\n");
    vider(lptr1, 0);
    vider(lptr2, 0);

    printf("%-60s : ", "-Egalite de deux listes vides");
    afficher(lptr1, FORWARD);
    printf(" et ");
    afficher(lptr2, FORWARD);
    printf(" %16s %d", "=", sontEgales(lptr2, lptr1));
    printf("\n");

    for (Info i = 0; i < 4; ++i)
        insererEnTete(lptr2, &i);
    for (Info i = 0; i < 4; ++i)
        insererEnTete(lptr1, &i);
    printf("%-60s : ", "-Egalite de deux listes de memes infos et meme ordre");
    afficher(lptr1, FORWARD);
    printf(" et ");
    afficher(lptr2, FORWARD);
    printf(" %2s %d", "=", sontEgales(lptr2, lptr1));
    printf("\n");

    vider(lptr1, 0);
    for (Info i = 4; i > 0; --i)
        insererEnTete(lptr1, &i);
    printf("%-60s : ", "-Egalite de deux listes de memes infos et d'ordre different");
```

```
    afficher(lptr1, FORWARD);
    printf(" et ");
    afficher(lptr2, FORWARD);
    printf(" %2s %d", "=", sontEgales(lptr2, lptr1));
    printf("\n");

    vider(lptr2, 2);
    printf("%-60s : ", "-Egalite de deux listes de differente taille");
    afficher(lptr1, FORWARD);
    printf(" et ");
    afficher(lptr2, FORWARD);
    printf(" %6s %d", "=", sontEgales(lptr2, lptr1));
    printf("\n");

    printf("\n----------### SUPPRIMER SELON CRITERE ###----------\n");

    printf("\nLes criteres qui permette la suppression utilises pour la fonction "
             "critere sont les suivant :\n"
             "-La position doit etre impaire ou la valeur doit se trouver dans "
             "l'intervalle ]1,6[.\n");

    tailleListe = 10;
    Liste *listeTestSupprimerSelonCritere = initialiser();

    for (Info i = 0; i < tailleListe; ++i)
        insererEnQueue(listeTestSupprimerSelonCritere, &i);
    printf("\n%s %d %-12s : ", "-Creation d'une liste de", tailleListe, "elements");
    afficher(listeTestSupprimerSelonCritere, FORWARD);
    printf("\n");

    supprimerSelonCritere(listeTestSupprimerSelonCritere, critere);

    printf("%-40s : ", "-La liste apres supprimerSelonCritere()");
    afficher(listeTestSupprimerSelonCritere, FORWARD);
    printf("\n");

    return 0;
}
```