```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

struct cellule {
    char* mot;
    /* ou bien char mot[512]; mais c'est moins propre */
    int nb;
    struct cellule * suiv;} ;

typedef struct cellule* LISTE;

LISTE creer_liste(void) {   return NULL;    }

int est_vide_Liste(LISTE L) {   return !L;  }

LISTE ajout_tete_Liste(char* m, LISTE L){
    LISTE p=(LISTE) calloc(1,sizeof(*p));
    if (p!=NULL) {
        p->mot=strdup(m);
        /* Ou bien strcpy(p->mot,m); si mot est definit par char mot[512]; */
        p->nb=1;
        p->suiv=L;
    }
    return p;
}

LISTE recherche_Liste(char* m, LISTE L){ LISTE p=L;
    while(!est_vide_Liste(p)&&(strcasecmp(p->mot,m)!=0)) p=p->suiv;
    return p;
}

void affiche_Liste(LISTE L){ LISTE p=L;
    while(!est_vide_Liste(p)) {
        printf("Le mot %s apparait %d fois\n",p->mot,p->nb);
        p=p->suiv;
    }
}


LISTE ajout1( char* m, LISTE histo) { LISTE p=NULL;
    p=recherche_Liste(m,histo);
    if (p==NULL) return ajout_tete_Liste(m,histo);
    else {
        p->nb++;
        return histo;
    }
}

LISTE libere(LISTE l) {LISTE p=l,c;
    for (p=l; !est_vide_Liste(p); p=c) {
        c=p->suiv;
        free(p->mot);
        free(p);
  }
    return NULL;
}
```

```
LISTE histogramme1(char* fic) { char s[512];
    FILE* f;
    LISTE histo=creer_liste();
    if ( (f=fopen(fic,"r"))==NULL) return NULL;
    while (fscanf(f,"%s",s)==1) {
        histo=ajout1(s,histo);
    }
    fclose(f);
    return histo;
}

typedef LISTE* TABLE;

int hachage(unsigned char* mot, int n) {
    int i,base=31;
    unsigned int u=0;
    for (i=strlen(mot)-1; i>=0; i--) u=(tolower(mot[i])+u*31)%n;
    return u;
}

TABLE ajout2( char* m, TABLE histo, int n) {
    int h=hachage(m,n);
    histo[h]=ajout1(m,histo[h]);
    return histo;
}

TABLE creer_Table(int n) {TABLE p;
    p=(TABLE) calloc(n,sizeof(*p));
    if (p!=NULL) { int i;
        for(i=0; i<n;i++) p[i]=creer_liste();
    }
    return p;
}


TABLE histogramme2(char* fic, int n) { char s[512];
    FILE* f;
    TABLE histo=creer_Table(n);
    if ( histo==NULL || (f=fopen(fic,"r"))==NULL) return NULL;
    while (fscanf(f,"%s",s)==1) {
        histo=ajout2(s,histo,n);
    }
    fclose(f);
    return histo;
}

void affiche_Table(TABLE L, int n){ int i;
    for(i=0; i<n; i++)
        if (!est_vide_Liste(L[i])) affiche_Liste(L[i]);
}

TABLE libere_Table(TABLE L, int n){ int i;
    for(i=0; i<n; i++)
        L[i]=libere_Liste(L[i]);
    free(L);
    return NULL;
}


LISTE ajout_trie(char* mot, LISTE l) { LISTE c;
```

```
  if (est_vide_Liste(l) || strcasecmp(mot,l->mot)<0) return ajout_tete_Liste
      (mot,l);
  else {
    for(c=l; c->suiv!=NULL && strcasecmp(mot,c->suiv->mot)>0; c=c->suiv) ;
    LISTE p=(LISTE) calloc(1,sizeof(*p));
    if (p!=NULL) {
      p->mot=strdup(mot);
                /* Ou bien strcpy(p->mot,m); si mot est definit par char
                    mot[512]; */
      p->nb=1;
      p->suiv=c->suiv;
      c->suiv=p;
    }
    return l;
  }
}

LISTE recherche_trie1(char* m, LISTE L){ LISTE p=L;
       while(!est_vide_Liste(p)&&(strcasecmp(p->mot,m)<0)) p=p->suiv;
       return (p && strcasecmp(p->mot,m)==0) ? p : NULL;
}

LISTE recherche_trie2(char* m, LISTE L){
       if (est_vide_Liste(L)) return NULL;
       else { int c= strcasecmp(L->mot,m);
         if (c==0) return L;
         else if (c>0) return NULL;
           else return recherche_trie2(m,L->suiv);
  }
}

LISTE ajout3( char* m, LISTE histo) { LISTE p=NULL;
       p=recherche_trie2(m,histo);
       if (p==NULL) return ajout_trie(m,histo);
       else {
               p->nb++;
               return histo;
       }
}

LISTE histogramme3(char* fic) { char s[15];
       FILE* f;
       LISTE histo=creer_liste();
       if ( (f=fopen(fic,"r"))==NULL) return NULL;
       while (fscanf(f,"%s",s)==1) {
               histo=ajout3(s,histo);
       }
       fclose(f);
       return histo;
}


main(int argc, char** argv) {
    LISTE h1=NULL;
    TABLE h2=NULL;
    LISTE h3=NULL;
    h1=histogramme1(argv[1]);
    puts("Histo par listes");
    affiche_Liste(h1);
    h2=histogramme2(argv[1],1000);
```

```
        puts("Histo par table");
        affiche_Table(h2,1000);
        h3=histogramme3(argv[1]);
        puts("Histo par liste triee");
        affiche_Liste(h3);
    }
```