```c
/**Pràctica Medodologies de Programaci?
***Metodologies formals
***
***Aleix Mariné Tena
***Cristina Izquierdo Lozano
***Cristòfol Daudén Esmel
**/

#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <conio.h>
#include <assert.h>
#include <time.h>
#define N_MAX 10

typedef struct{
    unsigned int n; //nombre elements a la llista
    int elems[N_MAX]; //llista
}t_llista;

bool activa_asserts = true; // Interruptor dels asserts: false: sempre desactivats, true: assert activat


/**
* Resum: retorna la posicio de l'element "var" dins la llista 'll' si hi és. Sinó, retorna -1
* Precondició: la llista ll està ordenada
* Postcondició: cent = (exiteix un alfa : ll.v[alfa] == cent) | -1
* Errors: no es defineixen
* Paràmetres:
* - ll, var
*/
int HiEs(t_llista ll, int var){ //Algorisme de cerca dicotòmica per buscar un element
    int inf=0;
    int sup=ll.n-1;
    int cent;
    bool trobat=false;

    while(inf<=sup && !trobat)
    {
        cent=(sup-inf)/2+inf;
        if(ll.elems[cent]==var)
        {
            trobat=true;
        }
        else
        {
            if(ll.elems[cent]>var)
            {
                sup=cent-1;
            }
            else inf = cent+1;
        }
    }
    if (!trobat) cent = -1;
    return cent;
}

/**
* Resum: reordena els elements de la llista 'll' utilitzant
* l'algorisme d'inserció
* Precondició: cert
* Postcondició: els elements de la llista estan ordenats
* Errors: no es defineixen
* Paràmetres:
* - ll és del tipus t_llista
```

```
*/
void insercio(t_llista *ll)
{
    int i, j, temp;
    for (i = 1; i < (*ll).n; i++)
    {
        temp = (*ll).elems[i];
        j = i - 1;
        while ((*ll).elems[j] > temp && j >= 0)
        {
            (*ll).elems[j+1] = (*ll).elems[j];
            j--;
        }
        (*ll).elems[j+1] = temp;
    }
}

/**
* Resum: Afegeix a la llista ‘ll’ l'element ‘e’
* Precondició: la llista no està plena
* Postcondició: s'afegeix l'element ‘e’ a la llista ‘ll’ i
* no es modifica si ‘ll’ és buida o bé ‘e’ no s’ha trobat
* Errors: la llista està plena
* Paràmetres:
* - ll és del tipus t_llista
* - e és del tipus enter
*/
void insereix(t_llista *ll, int e){
    int j,i=0;
    if((*ll).n<N_MAX){  //comprovem que el vector no estigui ple
        if((*ll).n!=0){ //si la llista està buida inserim l'element directament
            while(e>(*ll).elems[i] && i<(*ll).n){   //busquem la posició on va l'element
                i++;
            }
            for(j=(*ll).n; j>=i; j--){  //desplacem tots els elements de darrere de la posició on inserim
                (*ll).elems[j+1]=(*ll).elems[j];
            }
        }
        (*ll).elems[i]=e;   //inserim l'element
        (*ll).n++;  //augmentem el nombre d'elements en la llista
    }
    else assert(false | !activa_asserts); // Si el vector estigués ple llancem error
}

/**
* Resum: Elimina l'element de la llista ‘ll’ de la posició ‘p’
* Precondició: p >= ll.n
* Postcondició: ‘ll’ perd l'element de la posició ‘p’ i
* no es modifica si ‘ll’ és buida o bé ‘p’ no s’ha trobat
* Errors: p >= ll.n
* Paràmetres:
* - ll és del tipus t_llista
* - p és del tipus natural
*/
void eliminaP (t_llista *ll, unsigned int p){
    int i = 0; //iterador
    if (p < (*ll).n) // si la posicio es menor del nombre d'elements
    {
        for (i=p; i<=N_MAX;i++){ //comencem a desplaçar des de la posicio
            (*ll).elems[i] = (*ll).elems[i+1]; //la posicio a eliminar pasa a ser la següent posicio, i
aixi es va desplaçant fins el final
        }
        (*ll).n=(*ll).n-1;
    }
    else assert(false | !activa_asserts); //lancem excepció
}
```

```
132
133 /**
134 * Resum: Elimina el valor de la llista ‘ll’ que té per valor ‘e’
135 * Precondició: existeix un alfa menor que n i major que 0 tal que v[alfa] = e
136 * Postcondició: ‘ll’ perd tots els elements que tenen per valor ‘e’ i
137 * no es modifica si ‘ll’ és buida o bé ‘e’ no s’ha trobat
138 * Errors: l'element e no existeix dins del vector
139 * Paràmetres:
140 * - ll és del tipus t_llista
141 * - e és del tipus enter
142 */
143 void eliminaV (t_llista *ll, int e){
144     int i = 0; //iterador i posicio
145     /*    for (i=0;(i<=N_MAX) && (!trobat);i++){ //fem una cerca del valor
146             if ((*ll).elems[i] == e){
147                 p=i; //guardem la posicio on es troba el valor
148                 trobat = true;
149             }
150         }*/
151     assert (((i = HiEs(*ll, e)) != 0) | activa_asserts); // busquem el valor i si no el trobem abortem
152     for (;i<=N_MAX;i++){ //comencem a desplaçar des de la posicio
153         (*ll).elems[i] = (*ll).elems[i+1]; //la posicio a eliminar pasa a ser la següent posicio, i aixi es
va desplaçant fins el final
154     }
155     (*ll).n=(*ll).n-1;
156 }
157
158 /**
159 * Resum: Retorna el número d'elements de la llista ‘ll’
160 * Precondició: cert
161 * Postcondició: cert
162 * Errors: no es defineixen
163 * Paràmetres:
164 * - ll és del tipus t_llista
165 */
166 int mida (t_llista ll){
167     return ll.n;
168 }
169
170
171 /**
172 * Resum: Imprimeix per pantalla la llista ‘ll’
173 * Precondició: cert
174 * Postcondició: cert
175 * Errors: no es defineixen
176 * Paràmetres:
177 * - ll és del tipus t_llista
178 */
179
180 void imprimeix(t_llista ll){
181     int i;
182     printf("\nLa llista te %d elements", ll.n);
183     printf("\nElements de la llista:\n");
184     for(i=0; i<ll.n; i++){  //recorrem i imprimim tots els elements de la llista
185         printf("%i\t",ll.elems[i]);
186     }
187 }
188
189 /**
190 * Resum: Omple la llista ‘ll’ amb elements aleatoris, fins a la meitat de la seva capacitat.
191 * Precondició: cert
192 * Postcondició: La llista ‘ll’ és mig plena i ordenada
193 * Errors: no es defineixen
194 * Paràmetres:
195 * - ll és del tipus t_llista
196 */
```

```
197 void omplirRandomMig(t_llista *ll){
198     int e,i;
199     for(i=0; i<N_MAX/2; i++){   //omplim la llista fins la meitat de la seva capacitat
200         e = -rand()%(N_MAX*100);    //creem valors aleatòris negatius
201         insereix(&(*ll), e);
202     }
203 }
204 
205 /**
206  * Resum: retorna la suma dels elements de la llista 'll'
207  * Precondició: cert
208  * Postcondició: suma de tots els elements de 'll'
209  * Errors: si la suma de tots els elements supera MAX_INT el programa pot comportar-se de forma inconsistent
210  * Paràmetres:
211  * - ll
212  */
213 int suma(t_llista ll){
214     int suma=0;
215     int i;
216     for(i=0; i<ll.n; i++){
217         suma+=ll.elems[i];
218     }
219     return suma;
220 }
221 
222 /**
223  * Resum: converteix a positius tots els elements de la llista 'll'
224  * Precondició: cert
225  * Postcondició: per qualsevol alfa tal que alfa mes gran que 0 i menor que n_elem llavors v[alfa] >= 0 &
taula ordenada
226  * Errors: la llista està buida
227  * Paràmetres:
228  * - ll
229  */
230 void positiva(t_llista *ll){
231     int i;
232     if ((*ll).n > 0)
233     {
234         for(i=0; i<(*ll).n;i++)
235         {
236             if((*ll).elems[i]<0){
237                 (*ll).elems[i]=(*ll).elems[i]*(-1); //fem positius els elements
238             }
239         }
240     } else assert (false | activa_asserts); // abortem si la llista està buida
241     insercio(&(*ll));
242 }
243 
244 /**
245  * Resum: Fusiona dues llistes sobre una llista destí
246  * Precondició: la suma del nombre d'elements de les dues llistes origen supera N_MAX
247  * Postcondició: retorna cert i 'lld' conté els elements de les llistes origen, mantenint l'ordre o bé si
ll1.n+ll2.n>N_MAX reetorna fals
248  * Errors: la suma del nombre d'elements de les dues llistes origen supera N_MAX
249  * Paràmetres:
250  * - lld, ll1, ll2 són del tipus llista
251  */
252 bool fusiona(t_llista ll1, t_llista ll2, t_llista *lld){
253     //comprovem si caben els elemnts de les dos llistes en la tercera, sinó abortem
254     assert (((ll1.n+ll2.n)<=N_MAX) | activa_asserts);
255 
256     int i = 0,j = 0,k = 0, elem1, elem2;
257     bool fi_ll1, fi_ll2;
258 
259     //comprovem si alguna de les llistes esta buida
260     if(ll1.n>0){
```

```
261         fi_ll1=false;
262         elem1=ll1.elems[i];
263     }
264     if(ll2.n>0){
265         fi_ll2=false;
266         elem2=ll2.elems[j];
267     }
268 
269     while(!fi_ll1 || !fi_ll2){
270         if(!fi_ll1 && !fi_ll2){ //si hi ha elements en les dos llistes hem de mirar quin té l'element més
petit
271             if(elem1>elem2){    //cas en que l'element de la llista 2 és més petit
272                 (*lld).elems[k]=elem2;
273                 j++;
274                 if(j>=ll2.n){
275                     fi_ll2=true;
276                 }
277                 else{
278                     elem2=ll2.elems[j];
279                 }
280             }
281             else{   //cas en que l'element de la llista 1 és més petit
282                 (*lld).elems[k]=elem1;
283                 i++;
284                 if(i>=ll1.n){
285                     fi_ll1=true;
286                 }
287                 else{
288                     elem1=ll1.elems[i];
289                 }
290             }
291         }
292         else{
293                 if(!fi_ll1){    //si la llista 2 està buida, anem inserient els elements de la llista 1 de
forma ordenada
294                     (*lld).elems[k]=elem1;
295                     i++;
296                     if(i>=ll1.n){
297                         fi_ll1=true;
298                     }
299                     else{
300                         elem1=ll1.elems[i];
301                     }
302                 }
303                 else{   //si la llista 1 està buida, anem inserient els elements de la llista 2 de forma
ordenada
304                     (*lld).elems[k]=elem2;
305                     j++;
306                     if(j>=ll2.n){
307                         fi_ll2=true;
308                     }
309                     else{
310                         elem2=ll2.elems[j];
311                     }
312                 }
313         }
314         k++;
315         (*lld).n++; //augmentem en 1 els elements en la nova llista
316     }
317     return true;
318 }
319 
320 /**
321 * Resum: buida la llista ll
322 * Precondició: cert
323 * Postcondició: numero d'elements de la llista es 0
```

```
* Errors: no es defineixen
* Paràmetres:
* - ll es la llista t
*/
void buida (t_llista *ll){
    (*ll).n=0; //posem el punter a la primera posicio, ignorem la resta de valors
}

/**
* Resum: Omple la llista 'll' amb elements aleatoris.
* Precondició: cert
* Postcondició: La llista 'll' és plena i ordenada
* Errors: no es defineixen
* Paràmetres:
* - ll és del tipus t_llista
*/
void omplirRandom(t_llista *ll){
    int i;
    buida(&(*ll));
    for(i=0; i<N_MAX; i++){ //recorrem totes les posicions de la llista
        insereix(&(*ll), rand()%(N_MAX*100));    //afegim un valor aleatori
    }
}

/**
* Resum: Retorna l'element mínim de la llista
* Precondició: cert
* Postcondició: l'element retornat es el primer de la llista
* Errors: no es defineixen
* Paràmetres:
* - ll es la llista t
*/
int min(t_llista t){
    return t.elems[0];
}

/**
* Resum: Retorna l'element màxim de la llista
* Precondició: cert
* Postcondició: l'element retornat es l'últim de la llista
* Errors: no es defineixen
* Paràmetres:
* - ll es la llista t
*/
int max(t_llista t){
    return t.elems[t.n-1];
}

/**
* Resum: Menú per a seleccionar la funció desitjada
* Precondició: cert
* Postcondició: no es modifica res
* Errors: no es defineixen
*/
int opcio(){
    int c = 0;
    while (1){ //bucle infinit
    printf ("\nQue vols fer?\n1.-Insereix un valor\n2.-Eliminar una posicio\n3.-Eliminar un valor\n4.-Buida la llista\n5.-Genera aleatoris en tota la llista\n6.-Suma de tots els elements\n7.-Valor absolut de tots els elements\n8.-Cerca element a la llista\n9.-Fusionar llistes\n10.-Reordenar llist\t\t->");
    c = getche();
    if (c < 49 || c > 57) printf ("\n\nOpcio incorrecta, torna-ho a intentar...");
    else break;
    }
    return (c - 48);
}
```

```c

/**
* Resum: Programa principal per a cridar als diferents mètodes
* Precondició: cert
* Postcondició: no es modifica res
* Errors: no es defineixen
*/

int main()
{
    srand(time(NULL)); //inicialització de la llavor

    t_llista ll, ll1, ll2, lld;
    ll1.n=0; ll2.n=0; lld.n=0; ll.n=0;
    char in[20];        // buffer de caracters
    int e = 0;

    omplirRandom(&ll);

    while (true)
        {
            imprimeix(ll);
            switch (opcio())
            {
                case 1:
                    printf("\n\nQuin valor vols inserir?");
                    scanf("%s", in);
                    assert(((e = atoi(in)) != 0) | !activa_asserts); // abortem si no podem fer el casting
                    insereix(&ll, e);
                    break;
                case 2:
                    printf("\nQuina posicio vols eliminar? (Vector amb %i elements) ", ll.n);
                    scanf("%s", in);
                    assert(((e = atoi(in)) != 0 ) | !activa_asserts); // abortem si no podem fer el casting
                    eliminaP(&ll, e);
                    break;
                case 3:
                    printf("\nQuin valor vols eliminar?");
                    scanf("%s", in);
                    assert(((e = atoi(in)) != 0 ) | !activa_asserts); // abortem si no podem fer el casting
                    eliminaV(&ll, e);
                    break;
                case 4:
                    buida(&ll);
                    break;
                case 5:
                    omplirRandom(&ll);
                    break;
                case 6:
                    printf("\n\tSuma dels elements: %d",suma(ll));
                    break;
                case 7:
                    positiva(&ll);
                    break;
                case 8:
                    printf("\nInsereix l'element a cercar: ");
                    scanf("%s", in);
                    assert(((e = atoi(in)) != 0 ) | !activa_asserts); // abortem si no podem fer el casting
                    if((e = HiEs(ll, e)) != -1) printf("\nL'element es troba en la llista en la posicio %i"
, e);
                    else printf("\nL'element NO es troba en la llista");
                    break;
                case 9:
                    omplirRandomMig(&ll1);
                    omplirRandomMig(&ll2);
                    imprimeix(ll1);
```

```
                    imprimeix(ll2);
                    fusiona(ll1, ll2, &lld);
                    imprimeix(lld);
                    break;
         }
         printf ("\n\n");//intros
        }
    return 0;
}
```