Programming Assignment 2: Linear List

This assignment is to implement a Dev-C++ project of ***non-ordered*** and ***non-duplicate*** linear lists using single linked list. The specification of the linear list using single linked list is defined as the following data structure:

```
typedef int ElemType; // Define the linear list element type as an integer.

typedef struct node { // Define linked list node
   ElemType elem; // Node data, integer
   struct node* next; // Node link, defined recursively
} Node; // Node type

typedef Node* Link; // Node link, pointer to a node.

// The ordered linear list is a node pointer, pointing to the
// head node of the linked list.
typedef Link List;
```

The followings are function specification of List operations:

```
// Initialize a linear list, set its size to 0.
void initial(List *);

// The length of a linear list, returns the number of elements, namely size.
int getSize(List);

// Get the element at a position from a linear list, return the designated element.
ElemType getElem(List , int);

// Set the element at a position in a linear list to a specific element.
ElemType setElem(List L, ElemType, int);

// Find the position of an element in L. If successful, return the position of the
// element; otherwise, returns -1.
int search(List, ElemType);

// Insert an element after the end of a linear list, return the position of the
// inserted element. If the inserted element exists in the linear list or the linear list
// is overflow, insertElem() fails, and returns -1.
int insertElem(List *, ElemType);

// Delete an element from a list. If the element is in the linear list, delete it and
// return its position; otherwise, return -1.
int deleteElem(List *, ElemType);

// Print all elements of the linear list starting from the head.
int printList(List);
```

Assume the linear list is an abstract data type such that its implementation is hidden from the application programmer. That is, the main program cannot access the data structure of linear lists directly. In the main program implement the following three functions:

```
// Append list L2 at the end of L1. If an element of L2 exists in L1, discard that
// element. Return the result of append() operation.
List append(List L1, List L2);

// Join two lists L1 and L2. Return a list containing all common elements in L1
// and L2. The elements of the resulting list are stored in the order of list L1.
```

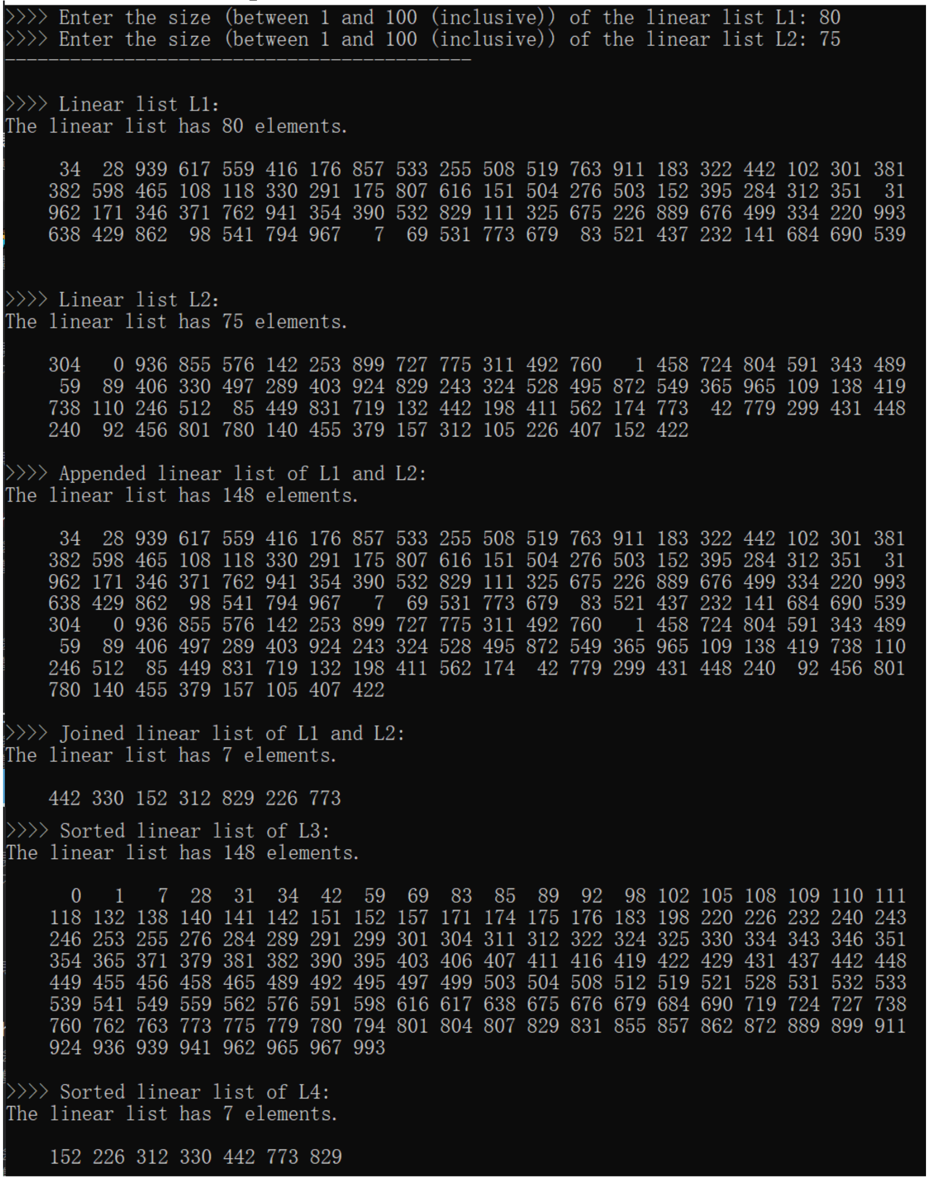
List join(List L1, List L2)

// Sort list L. The elements of L are rearranged into the ascending order.
void sort(List *L);

Write a C program to perform the following steps:

(1) Declare two lists L1 and L2.
(2) Enter two positive integer n1 and n2 such that 20≤n1, n2≤200, and insert n1 and n2 elements of random number between 0 and 999 to linear list L1 and L2, respectively. No duplicated elements are allowed in a linear list.
(3) Print the elements of linear lists L1 and L2.
(4) Compute and print L3 as the result of append(L1, L2).
(5) Compute and print L4 as the result of join(L1, L2).
(6) Sort L3 and print L3 after sorting.
(7) Sort L4 and print L4 after sorting.

In this assignment, you must submit five files: project, header, and source files **assgn2_DXXXXXXX.dev**, **assgn2_DXXXXXXX.h**, **assgn2_DXXXXXXX.c**, and **assgn2_DXXXXXXX_main.c** (80%) and the assignment report **assgn2_DXXXXXXX.pdf** (20%), where DXXXXXXX is your student ID. In the assignment report, you should explain how image rotation is performed. The assignment is due by **23:59 pm, Sunday, March 19, 2023**. Program execution example:



```
>>>> Enter the size (between 1 and 100 (inclusive)) of the linear list L1: 80
>>>> Enter the size (between 1 and 100 (inclusive)) of the linear list L2: 75
-------------------------------------------

>>>> Linear list L1:
The linear list has 80 elements.

     34  28 939 617 559 416 176 857 533 255 508 519 763 911 183 322 442 102 301 381
    382 598 465 108 118 330 291 175 807 616 151 504 276 503 152 395 284 312 351  31
    962 171 346 371 762 941 354 390 532 829 111 325 675 226 889 676 499 334 220 993
    638 429 862  98 541 794 967   7  69 531 773 679  83 521 437 232 141 684 690 539


>>>> Linear list L2:
The linear list has 75 elements.

    304   0 936 855 576 142 253 899 727 775 311 492 760   1 458 724 804 591 343 489
     59  89 406 330 497 289 403 924 829 243 324 528 495 872 549 365 965 109 138 419
    738 110 246 512  85 449 831 719 132 442 198 411 562 174 773  42 779 299 431 448
    240  92 456 801 780 140 455 379 157 312 105 226 407 152 422

>>>> Appended linear list of L1 and L2:
The linear list has 148 elements.

     34  28 939 617 559 416 176 857 533 255 508 519 763 911 183 322 442 102 301 381
    382 598 465 108 118 330 291 175 807 616 151 504 276 503 152 395 284 312 351  31
    962 171 346 371 762 941 354 390 532 829 111 325 675 226 889 676 499 334 220 993
    638 429 862  98 541 794 967   7  69 531 773 679  83 521 437 232 141 684 690 539
    304   0 936 855 576 142 253 899 727 775 311 492 760   1 458 724 804 591 343 489
     59  89 406 497 289 403 924 243 324 528 495 872 549 365 965 109 138 419 738 110
    246 512  85 449 831 719 132 198 411 562 174  42 779 299 431 448 240  92 456 801
    780 140 455 379 157 105 407 422

>>>> Joined linear list of L1 and L2:
The linear list has 7 elements.

    442 330 152 312 829 226 773

>>>> Sorted linear list of L3:
The linear list has 148 elements.

      0   1   7  28  31  34  42  59  69  83  85  89  92  98 102 105 108 109 110 111
    118 132 138 140 141 142 151 152 157 171 174 175 176 183 198 220 226 232 240 243
    246 253 255 276 284 289 291 299 301 304 311 312 322 324 325 330 334 343 346 351
    354 365 371 379 381 382 390 395 403 406 407 411 416 419 422 429 431 437 442 448
    449 455 456 458 465 489 492 495 497 499 503 504 508 512 519 521 528 531 532 533
    539 541 549 559 562 576 591 598 616 617 638 675 676 679 684 690 719 724 727 738
    760 762 763 773 775 779 780 794 801 804 807 829 831 855 857 862 872 889 899 911
    924 936 939 941 962 965 967 993

>>>> Sorted linear list of L4:
The linear list has 7 elements.

    152 226 312 330 442 773 829
```