Section A – Linked Lists

1. **(insertSortedLL)** Write a C function `insertSortedLL()` that asks the user to input an integer, then inserts it into the linked list in ascending order. The function, `insertSortedLL()`, should not allow inserting an integer if it already exists in the current linked list. The function should return the index position where the new item was added; if the function could not complete successfully, it should return a value of -1. You can assume that the linked list is either a sorted linked list or an empty list. *ListNode *temp;* *遍历条件是curindex <= ll->index, 走到底直接* *temp = ll->head;* *塞, 在途中每个比较有三种情况。*

   The function prototype is given as follows:
   ```
   int insertSortedLL(LinkedList *ll, int item);
   ```

   If the current linked list is: **2, 3, 5, 7, 9.**
   Calling `insertSortedLL()` with a value of **8** will result in the following linked list:
   2, 3, 5, 7, 8, 9.

   The function should return the index position where the new item was added as follows:
   The value 8 was added at index 4

   If the current linked list is: **5, 7, 9, 11, 15.**
   Calling `insertSortedLL()` with a value of **7** will result in the following linked list:
   5, 7, 9 , 11, 15.

   The function does not complete successfully (does not insert the value of 7 to the linked list) hence it should return a value of -1:
   The value 7 was added at index -1

   Some sample inputs and outputs are given as follows:
   ```
   1: Insert an integer to the sorted linked list:
   2: Print the index of the most recent input value: 3: Print sorted
   linked list:
   0: Quit:

   Please input your choice(1/2/3/0): 1
   Input an integer that you want to add to the linked list: 2
   The resulting linked list is: 2

   Please input your choice(1/2/3/0): 1
   Input an integer that you want to add to the linked list: 3
   The resulting linked list is: 2 3

   Please input your choice(1/2/3/0): 1
   Input an integer that you want to add to the linked list: 5
   ```

```
The resulting linked list is: 2 3 5

Please input your choice(1/2/3/0): 1
Input an integer that you want to add to the linked list: 7
The resulting linked list is: 2 3 5 7

Please input your choice(1/2/3/0):1
Input an integer that you want to add to the linked list: 9
The resulting linked list is: 2 3 5 7 9

Please input your choice(1/2/3/0): 1
Input an integer that you want to add to the linked list: 8
The resulting linked list is: 2 3 5 7 8 9

Please input your choice(1/2/3/0): 2
The value 8 was added at index 4

Please input your choice(1/2/3/0):3
The resulting sorted linked list is: 2 3 5 7 8 9

Please input your choice(1/2/3/0): 1
Input an integer that you want to add to the linked list: 5
The resulting linked list is: 2 3 5 7 8 9

Please input your choice(1/2/3/0): 2
The value 5 was added at index -1

Please input your choice(1/2/3/0):3
The resulting sorted linked list is: 2 3 5 7 8 9

Please input your choice(1/2/3/0): 1
Input an integer that you want to add to the linked list: 11
The resulting linked list is: 2 3 5 7 8 9 11

Please input your choice(1/2/3/0): 2
The value 11 was added at index 6

Please input your choice(1/2/3/0):3
The resulting sorted linked list is: 2 3 5 7 8 9 11
```

2. **(alternateMergeLL)** Write a C function `alternateMergeLL()` which inserts nodes of the second list into alternate positions of the first list. The nodes of the second list should only be inserted when there are alternate positions available in the first list.

The function prototype is given as follows: 两个fixsize变量
```
void alternateMergeLL(LinkedList *ll1, LinkedList *ll2);
```

For an example, assume that given two linked lists are LinkedList1 and LinkedList2:
LinkedList1: **1, 2, 3**
LinkedList2: **4, 5, 6, 7**

The resulting linked lists are:
LinkedList1: **1, 4, 2, 5, 3, 6**
LinkedList2: **7**

The second list should become empty when the first list is larger than the second list. For an example, assume that given two linked lists are LinkedList1 and LinkedList2:
LinkedList1: **1, 5, 7, 3, 9, 11**
LinkedList2: **6, 10, 2, 4**

The resulting Linked Lists are:
LinkedList1: **1, 6, 5, 10, 7, 2, 3, 4, 9, 11**
LinkedList2: **empty**

A sample input and output session is given below (if the current linked lists are Linked list 1: **1, 2, 3** and Linked list 2: **4 , 5, 6, 7**):

```
Linked list 1: 1 2 3
Linked list 2: 4 5 6 7

Please input your choice(1/2/3/0): 3
The resulting linked lists after merging the given linked list are:
Linked list 1: 1 4 2 5 3 6
Linked list 2: 7
```

3. **(moveOddItemsToBackLL)** Write a C function `moveOddItemsToBackLL()` that moves all the odd integers to the back of the linked list.
    先找出odd number个数count，遍历count次每次放后一个，重要的是temp和count在每次遍历都要初始化
    The function prototype is given as follows:
    ```
    void moveOddItemsToBackLL(LinkedList *ll);
    ```

    Some sample inputs and outputs sessions are given below:
    If the linked list is **2, 3, 4, 7, 15, 18**:
    ```
    The resulting Linked List after moving odd integers to the back of the
    Linked List is: 2 4 18 3 7 15
    ```

    If the linked list is **2, 7, 18, 3, 4, 15**:
    ```
    The resulting Linked List after moving odd integers to the back of the
    Linked List is: 2 18 4 7 3 15
    ```

    If the current linked list is **1, 3, 5**:
    ```
    The resulting Linked List after moving odd integers to the back of the
    Linked List is: 1 3 5
    ```

    ```
    If the current linked list is 2 4 6:
    The resulting Linked List after moving odd integers to the back of the
    Linked List is: 2 4 6
    ```

4. **(moveEvenItemsToBackLL)** Write a C function `moveEvenItemsToBackLL()` that moves all the even integers to the back of the linked list.

   The function prototype is given as follows:
   ```
   void moveEvenItemsToBackLL(LinkedList *ll);
   ```

   Some sample inputs and outputs sessions are given below:
   If the linked list is **2, 3, 4, 7, 15, 18**:
   ```
   The resulting Linked List after moving even integers to the
   back of the Linked List is: 3 7 15 2 4 18
   ```

   If the linked list is **2, 7, 18, 3, 4, 15**:
   ```
   The resulting Linked List after moving even integers to the
   back of the Linked List is: 7 3 15 2 18 4
   ```

   If the current linked list is **1, 3, 5**:
   ```
   The resulting Linked List after moving even integers to the
   back of the Linked List is: 1 3 5
   ```

   ```
   If the current linked list is 2 4 6:
   The resulting Linked List after moving even integers to the
   back of the Linked List is: 2 4 6
   ```

5. **(frontBackSplitLL)** Write a C function `frontBackSplitLL()` that splits the singly linked list into two sublists – one for the front half, and one for the back half. If the number of elements is odd, the extra element should go into the front list. The `frontBackSplitLL()` prints two lists which contains `frontList` and `backList`.

   The function prototypes are given as follows:
   ```
   void frontBackSplitLL(LinkedList *ll, LinkedList
   *resultFrontList, LinkedList *resultBackList);
   ```
   fixsize = ll->size

   For example, assume that given linked list is: **2, 3, 5, 6, 7**

   The resulting Linked Lists, frontList and backList are:
   frontList: **2, 3, 5**
   backList: **6, 7**

   A sample input and output session is given below:
   ```
   1: Insert an integer to the linked list:
   2: Print the linked list:
   3: Split the linked list into two linked lists, frontList and backList:
   0: Quit:

   Please input your choice(1/2/3/0): 1
   ```

```
Input an integer that you want to add to the linked list: 2
The resulting linked list is: 2

Please input your choice(1/2/3/0): 1
Input an integer that you want to add to the linked list: 3
The resulting linked list is: 2 3

Please input your choice(1/2/3/0): 1
Input an integer that you want to add to the linked list: 5
The resulting linked list is: 2 3 5

Please input your choice(1/2/3/0): 1
Input an integer that you want to add to the linked list: 6
The resulting linked list is: 2 3 5 6

Please input your choice(1/2/3/0): 1
Input an integer that you want to add to the linked list: 7
The resulting linked list is: 2 3 5 6 7

Please input your choice(1/2/3/0): 2
The resulting linked list is: 2 3 5 6 7

Please input your choice(1/2/3/0): 3
The resulting linked lists after splitting the given linked list are:
Front linked list: 2 3 5
Back linked list: 6 7
```

6. **(moveMaxToFront)** Write a C function `moveMaxToFront()` that traverses a linked list of integers at most once, then moves the node with the largest stored value to the front of the list. 如果只有一个term，直接过；else 设三个listnode，一个遍历(遍完要初始化)，一个指最大，一个指最大前面；front->next == max->next; max->next = *ptrHead; *ptrHead = max;

The function prototype is given as follows:
```
int moveMaxToFront(ListNode **ptrHead);
```
这个函数头不同于之前：1.故Insertnode功能不可用 2.ListNode *temp；temp = *ptrHead；

For example, if the linked list is (**30, 20, 40, 70, 50**), the resulting linked list will be (**70, 30, 20, 40, 50**).

```
1: Insert an integer to the linked list:
2: Move the node with the largest stored value to the front of the
list:
0: Quit:

Please input your choice(1/2/0): 1
Input an integer that you want to add to the linked list: 30
The Linked List is: 30
Please input your choice(1/2/0): 1
Input an integer that you want to add to the linked list: 20
The Linked List is: 30 20
Please input your choice(1/2/0): 1
Input an integer that you want to add to the linked list: 40
The Linked List is: 30 20 40
```

```
Please input your choice(1/2/0): 1
Input an integer that you want to add to the linked list: 70
The Linked List is: 30 20 40 70
Please input your choice(1/2/0): 1
Input an integer that you want to add to the linked list: 50
The Linked List is: 30 20 40 70 50

Please input your choice(1/2/0): 2
The resulting Linked List is: 70 30 20 40 50

Please input your choice(1/2/0): 0
```

7. **(recursiveReverse)** Write a C function `recursiveReverse()` that recursively reverses the given linked list by changing its next pointer and its head pointer.

The function prototype is given as follows:
```
void recursiveReverse(ListNode **ptrHead);
```

For example, if the linked list is (**1, 2, 3, 4, 5**), the resulting linked list will be (**5, 4, 3, 2, 1**).

A sample input and output session is given below:
```
1: Insert an integer to the linked list:
2: Reversed Linked List:
0: Quit:
```

背，两个Listnode，*first，*rest，
如有NULL直接return；
recursive（&rest）；
first-next-next = first;
first-next = NULL;
*ptrHead = rest;

```
Please input your choice(1/2/0): 1
Input an integer that you want to add to the linked list: 1
The resulting Linked List is: 1
Please input your choice(1/2/0): 1
Input an integer that you want to add to the linked list: 2
The resulting Linked List is: 1 2
Please input your choice(1/2/0): 1
Input an integer that you want to add to the linked list: 3
The resulting Linked List is: 1 2 3
Please input your choice(1/2/0): 1
Input an integer that you want to add to the linked list: 4
The resulting Linked List is: 1 2 3 4
Please input your choice(1/2/0): 1
Input an integer that you want to add to the linked list: 5
The resulting Linked List is: 1 2 3 4 5

Please input your choice(1/2/0): 2
The resulting Linked List after reversing its elements is: 5 4 3 2 1

Please input your choice(1/2/0): 0
```