CE1007/CZ1007 – DATA STRUCTURES

Assignment Questions: Linked List

Information:

Program templates for questions 1-4 are given as separated files (**Q1_template_LL.c, Q2_template_LL.c**, Q3**_template_LL.c**, and Q4**_template_LL.c**). You **must use** them to implement your functions. The program contains a *main()* function, which includes a switch statement to execute different functions that you should implement. Each function can be called multiple times depending on the user's choice. You need to submit your code to the Automated Programming Submission and Assessment System (APAS) for testing.
Deadline for program submission: **October 28th, 2019 (Monday) 11.59 pm**.

Assignment Questions (1-4)

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.

   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: 5
   ```

```
The resulting linked list is: 2 5

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 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): 2
The value 7 was added at index 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 7

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

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: 9

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: 7 9

Please input your choice(1/2/3/0): 0
Press any key to continue . . .
```

2. **(moveEvenItemsToBackLL)** Write a C function `moveEvenItemsToBackLL()` that **moves all the even integers to the back of the linked list**. (**Note** : You may need to use and include the removeNode function done in your Lab/Tutorial)

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
```

3. **(appendLL)** Write a C function `appendLL()` that takes two lists, 'a' and 'b', appends 'b' onto the end of 'a', and then sets 'b' to NULL (since it is now trailing off the end of 'a').

   The function prototype is given below:

   ```
   void appendLL(LinkedList *ll_a , LinkedList *ll_b);
   ```

   For example, assume that two linked lists are as follows:
   Linked list a: 1 2 3 4 5
   Linked list a: 6 7 8 9 10
   The resulting linked list a: 1 2 3 4 5 6 7 8 9 10
   The resulting linked list b: Empty

   Sample test cases are given below:

   ```
   1: Insert an integer to the linked list a:
   2: Insert an integer to the linked list b:
   3: Merge two sorted linked lists:
   0: Quit:

   Please input your choice(1/2/3/0): 1
   Input an integer that you want to add to the linked list a: 1
   The resulting linked list a: 1
   Please input your choice(1/2/3/0): 1
   Input an integer that you want to add to the linked list a: 2
   The resulting linked list a: 1 2
   Please input your choice(1/2/3/0): 1
   Input an integer that you want to add to the linked list a: 3
   The resulting linked list a: 1 2 3
   Please input your choice(1/2/3/0): 1
   Input an integer that you want to add to the linked list a: 4
   The resulting linked list a: 1 2 3 4
   Please input your choice(1/2/3/0): 1
   Input an integer that you want to add to the linked list a: 5
   The resulting linked list a: 1 2 3 4 5
   Please input your choice(1/2/3/0): 2
   Input an integer that you want to add to the linked list b: 6
   The resulting linked list b: 6
   Please input your choice(1/2/3/0): 2
   Input an integer that you want to add to the linked list b: 7
   The resulting linked list b: 6 7
   Please input your choice(1/2/3/0): 2
   Input an integer that you want to add to the linked list b: 8
   The resulting linked list b: 6 7 8
   Please input your choice(1/2/3/0): 2
   Input an integer that you want to add to the linked list b: 9
   The resulting linked list b: 6 7 8 9
   Please input your choice(1/2/3/0): 2
   Input an integer that you want to add to the linked list b: 10
   The resulting linked list b: 6 7 8 9 10

   Please input your choice(1/2/3/0): 3
   The resulting linked lists after appending list b to list a are:
   The resulting linked list a: 1 2 3 4 5 6 7 8 9 10
   The resulting linked list b: Empty

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

4. **(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:

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
```