Due Date: March 29, Tuesday, 23:55 (submit online via ODTU-Class)

CENG 301

Spring 2015-2016

Assignment #1
Merging Two Linked Lists

In this assignment your goal is to write a function to merge two linked lists that contain integers in ascending order. The merged list should contain all the integers also in ascending order. The merging should be done in place meaning that you only arrange the links to merge the two lists. You are not allowed to create a new list or a new node. In other words in the mergeLists function that you are going to write, you are **not** allowed to call dynamic memory allocation functions such as *malloc*, *calloc*, and *realloc*.

Below is the C program that you are going to complete (you can also access this template at http://www.ceng.metu.edu.tr/~tcan/ceng301_s1516/Schedule/linkedList.c):

```
#include <stdio.h>

typedef struct Node
{
      int element;
      struct Node* next;
} Node;

Node* mergeLists(Node* list1, Node* list2)
{
   /* To be completed by you */
}

Node *createList(int x)
{
      Node *tmp;

      tmp =(Node *)malloc(sizeof(Node));
      tmp->element = x;
      tmp->next = NULL;

      return tmp;
}

Node* insertOrdered(Node *head, int x)
{
      Node *tmp;

      tmp = head;
```

```
	if (head->element > x) /* insert at beginning */
	{
		tmp = (Node *)malloc(sizeof(Node));
		tmp->element = x;
		tmp->next = head;
		return tmp;
	}

	while (tmp->next!=NULL && tmp->next->element <= x)
		tmp = tmp->next;

	if (tmp->next == NULL) /* insert at the end */
	{
		tmp->next = (Node *)malloc(sizeof(Node));
		tmp->next->element = x;
		tmp->next->next = NULL;

	}
	else /* insert in the middle */
	{
		Node *tmp2 = tmp->next;
		tmp->next = (Node *)malloc(sizeof(Node));
		tmp->next->element = x;
		tmp->next->next = tmp2;
	}
	return head;
}

void printList(Node *head)
{
	Node *tmp;
	tmp = head;
	while (tmp!=NULL)
	{
		printf("%d ",tmp->element);
		tmp = tmp->next;
	}
	printf("\n");

}

int main()
{
	Node *listA;
	Node *listB;
	Node *merged;

	listA = createList(6);
	listB = createList(4);

	listA = insertOrdered(listA,3);
```

```
      listA = insertOrdered(listA,9);
      listA = insertOrdered(listA,5);
      listA = insertOrdered(listA,10);
      listA = insertOrdered(listA,2);

      listB = insertOrdered(listB,1);
      listB = insertOrdered(listB,8);
      listB = insertOrdered(listB,7);
      listB = insertOrdered(listB,14);
      listB = insertOrdered(listB,11);
      listB = insertOrdered(listB,19);
      listB = insertOrdered(listB,15);


      printList(listA);
      printList(listB);

      merged = mergeLists(listA,listB);

      printList(merged);

}
```

If you implement the mergeLists function correctly, the output of the program should be:

```
2 3 5 6 9 10
1 4 7 8 11 14 15 19
1 2 3 4 5 6 7 8 9 10 11 14 15 19
```

Remember that you are NOT allowed to create a new node in your function, in other words do not use dynamic memory allocation functions such as *malloc*, *calloc*, and *realloc*. You should just arrange the links of the old lists to create the merged list. After this merging operation the original lists will be no longer available.

Submission:

Submit your source code named <student_id>.c via ODTU-Class before the deadline. Late submissions are allowed with 20 pts penalty per day.