Computer Science I – Spring 2012
Lab: Linked Lists, Part II

For each of these questions, use the following struct for a node in a linked list

```
struct ll
  int data;
  struct ll* next;
};
```

1) Write a function that takes in a pointer to the front of a linked list and returns 1 if all the nodes in the linked list are in sorted order (from smallest to largest, with repeats allowed), and 0 otherwise. The prototype is given below:

```
int inOrder(struct ll* list);
```

2) Write a function that takes in a pointer to the front of a linked list and adds a given number, *c*, to the data value in each node in the list. The prototype is given below:

```
int addc(struct ll* list, int c);
```

3) Write a function that takes in a pointer to the front of a linked list, a value to insert into the list, *val*, and a location in the list in which to insert it, *place*, which is guaranteed to be greater than 1, and does the insertion. If *place* number of items aren't in the list, just insert the item in the back of the list. You are guaranteed that the linked list into which the inserted item is being added is not empty. The prototype is given below:

```
void insertToPlace(struct ll* list, int val, int place);
```

4) (Aug 09) Write a function that operates on a linked list of integers. Your function should insert a new node containing the value 2 after every node that contains the value 4. Make use of the list node struct and function header below.

```
void list_42(struct ll* list);
```

5) (May 08) Write a function which returns a pointer to a linked list which is the result of moving the first node in the list pointed to by alpha to the end of the list. (For example, if alpha points to a list that contains 3, 7, 2, 1, and 8, then when the function is called, a pointer to a list that contains 7, 2, 1, 8, and 3 should be returned.) If alpha has zero or one nodes, just return a pointer to the original list without making any changes. Utilize the function prototype provided below:

```
struct node* moveFrontToBack(struct node* alpha);
```