## **Exercise Linked List**

For the following questions consider the following struct definition:

```
struct node{
   int data;
   struct node *next;
};
```

1. Write a function that takes a pointer to the head of a linked list and changes the list by adding an integer n (passed in as a parameter) to each node of the list.

```
void addN(struct node* list, int n){

while (list = NULL)

{ list > data = list > data + n;

list = list > next;}
```

**2.** Write a function that deletes the first node in a linked list and returns a pointer to the new head of the list. If there are no items in the original list, NULL should be returned.

```
struct node* deleteFirst(struct node* list) {

If (| st | = NULL) {

Node temp = | st;

| ist = | st > next;

free (temp); }

Peturn list;

}
```

**3.** Write a function that makes a copy of an input list and returns a pointer to it. Note: This function should call malloc once for each node in the original list.

```
struct node* copy(struct node* list) { node * NH = head. node * NL= NULL. node * NT= NULL.

while (NH = NULL) { If (NL == NULL)
```

**4.** p contains the elements 66, 9, 14, 52, 87, 14 and 17, in that order. Consider running the following line of code:

```
p = question4(p);
```

where question4 is the function defined below. Show the contents of p after the function call.

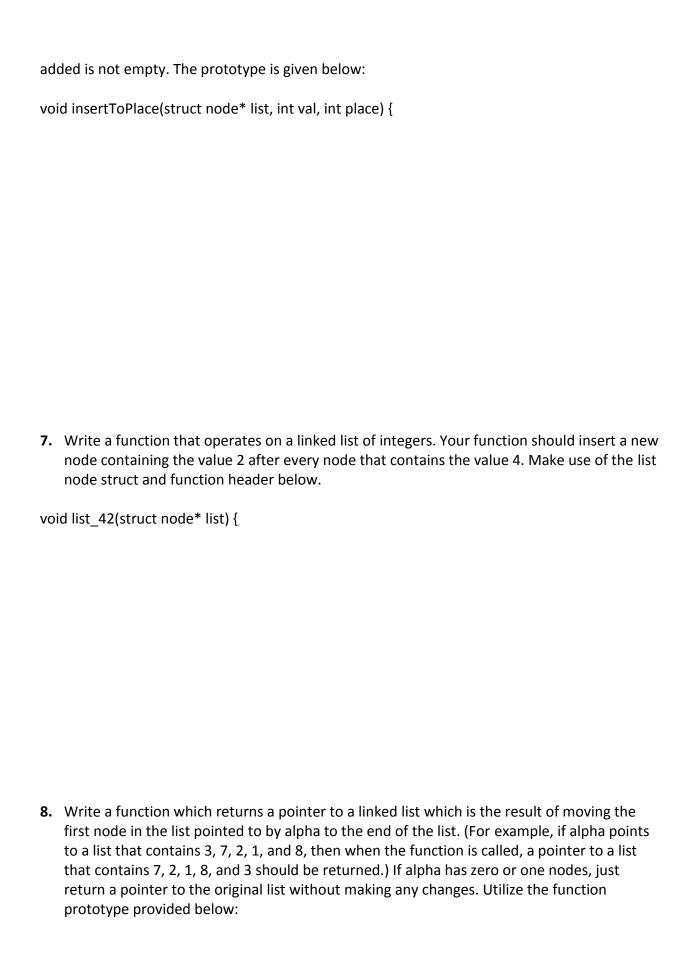
```
struct node* question4(struct node *list) {
    struct node* a = list;
    struct node* b = list;
    struct node* c;
    if (a == NULL) return NULL;
    while ( a->next != NULL)
        a = a ->next;
    a->next = b;
    c = b->next;
    b->next = NULL;
    return c;
}
```

Answer:

**5.** 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 isSorted(struct node\* list) {

**6.** Write a function that takes in a pointer to the head 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



	struct node* moveFrontToBack(struct node* alpha) {
9.	You want to implement stack using linked list. Push function will insert an item to the head in the linked list and pop() function will remove and return the head item from the linked list. Implement the following functions:
	The full code will be available in the weboucourses codes->linkedlist folder
10.	Implement Queue using linked list. [All the necessary steps are available in the slide]. We will try it at the lab]. You don't need to submit it with the exercise submission. You will have to submit this code as a lab submission by Friday. You can work on it in the lab.