THE UNIVERSITY OF SYDNEY

INFO1910 Week 10 Tutorial

In this tutorial you will work with linked lists and write functions to manage them. A linked list is made up of nodes or cells; each node contains an element of data (which in this case will be a single character) and a pointer to the next node in the list. By following pointers from the first node in the list, any element can be accessed. Use the following struct for linked list nodes.

```
struct ll_node {
    char elem;
    struct ll_node* next;
};
```

Nodes in the simplest kind of linked lists only have a pointer to the next node in the list, not the previous, and these are the kind we will work with in this tutorial. Linked lists with pointers in both directions are called 'doubly linked lists'.

Question 1

Write a function which will fill out a declared linked list node. The `next` pointer should be set to a NULL pointer.

```
void ll_node_create(struct ll_node* node, char elem)
```

Question 2

Write a function which, given a linked list and a node, will append that node on the end of the list. You will need to find the last node in the list and modify its `next` pointer. What is the condition that tells you the current node is the last one in the list?

```
void ll_append(struct ll_node* list, struct ll_node* new_node)
```

Question 3

Write a function which finds the length of a given linked list.

```
int ll_len(struct ll_node* list)
```

Question 4

Write a function which finds whether a particular element is present in a linked list. If so, it should return the index of the element's first occurence. If not, it should return -1.

```
int ll_contains(struct ll_node* list, char elem)
```

Question 5

Write a function which can retrieve the element at the given index in a linked list.

```
char ll_get(struct ll_node* list, int idx)
```

Question 6

Write a function which can insert a given node at a specific position in a linked list.

```
void ll_insert(struct ll_node* list, struct ll_node* new_node, int idx)
```

Question 7

Write a function which removes the first occurrence of a given element from the linked list, and returns it. Remember to update all the `next` pointers appropriately when removing nodes.

```
char ll_pop(struct ll_node* list, char elem)
```

Question 8

Write a function which takes in two linked lists, and returns a pointer to the head of a single linked list which contains all the elements from both; first the elements of list1, then the elements of list2.

```
struct ll_node* ll_extend(struct ll_node* list1, struct ll_node* list2)
```

Question 9

Test your linked list functions. You should now have most of the functionality of Python lists in your linked list function library; try doing some of the operations and idioms you are used to doing with Python lists.

Question 10

If you have been using a main function in the same source code file to test your functions, remove it. Create a header (.h) file declaring the existence of the functions in this file, and defining the linked list node struct. Practise calling these linked list functions from a separate program, and writing the makefile to compile it, so that the functions you have written can be used in other programs you may write in the future.