Linked List

The definition of a linked list is exactly the same as we covered last semester.

Every node in a linked list has two parts – a link to its data part and a link to the next node in the list (NULL if there is no next node). The list will have two external pointers – one to the first element and one to the last element.

Elements can be added to the front of the list, the end of the last and at any place in the middle of the list.-

```
//Stucture template for data part of a node
struct data {
  int num;
};

//Stucture template for one node
struct LinearNode {
        struct data *element;
        struct LinearNode *next;
};


struct LinearNode *front = NULL;
struct LinearNode *last = NULL;  [not in the code currently as I didn't need it but it can be easily added]
```

Add a new node to the end of the list

1. Create data part of the node

a. Declare a pointer variable which will hold the address of a data element

```
struct data *anElement;
```

b. Reserve space for the data part of the node

```
anElement = (struct data *)malloc(sizeof(struct data));
```

c. Add a value to this element

```
anElement->num  = aNumber;
```

2. Create space for the node

a. Declare a pointer variable which will hold the address of the node

```
struct LinearNode * aNode;
```

b. Reserve space for the node

```
aNode = (struct LinearNode *)malloc(sizeof(struct LinearNode));
```

3. Assign the data part created in step 1 above to this node

```
aNode->element = anElement;
```

4. Let its next part point to null

```
aNode->next = NULL;
```

5. Add node to the front of the list

If the list is empty, this new node will be the new front and last node

Otherwise let the next part of this new node point to this current front node and make this new node the new front node.

OR

6. Add node to the end of the list

If the list is empty, this new node will be the new front and last node

Otherwise let the next part of the last element point to this new node and make this the new last node.

Negotiate