A linked list is a sequence of nodes where a node consists of some data value and a link to the next node on the list. (The link field of the last node is NULL.)

The basic building block of a list is a node. In this example, we work relative to a node, called IntNode, whose data value is a single integer. A blueprint of the node follows.

Here is the class declaration for the node. It is based on the foregoing blueprint. (There are two constructors.)

```cpp
class IntNode
{
private:
    int data;
    IntNode* link;

public:
    IntNode(); //default constructor
    IntNode(int theData, IntNode* theLink); //Constructor with parameters
    void setData(int theData); //Post: data = theData
    void setLink(IntNode* theLink); //Post: link = theLink
    int getData() const; //Return data
    IntNode* getLink() const; //Return link
};
```
The following operations are to be implemented using *non-member functions*:

1. Check whether or not the linked list (presented with its head pointer) is empty.

2. Add a node holding a given data value at the head of the linked list, where the list itself is presented with its head pointer.

3. Return the length of a given list.

4. Insert a node holding a given data value somewhere in the middle of the linked list assuming that the pointer to an existing node (after which the new node is to be added) is known.

5. Delete the node that is positioned such that the pointer to the node to be removed and pointer to its immediately preceding node are known.

6. Determine the pointer to the node in the linked list (where the list is presented with its head pointer) such that the node being searched holds a given data value. If no such node exists, then return NULL.

7. Systematically print the data values of the nodes on the list.

8. Destroy a given list and return the memory to the free store.