This Lab Exercise involves designing and implementing a class called **`IntList`**, which creates a linked list of integer values. The general "policy" of this linked list is to maintain the list in sorted order. Notice, however, that the APPEND command does not honor that policy: it appends the new node to the end of the list, regardless of the data value.

Due Date

You must *submit* the source code for the solution to this lab exercise to *Moodle* by

Tuesday, August 12, 2025

in order to receive full credit for this work. You must also *demonstrate* the solution to the instructor during class, at the earliest opportunity.

Programming Assignment

Create **IntList.h** and **IntList.cpp** as specified below. Use the **NumberList** class (described in the textbook and available on *Moodle*) as a starting point for writing your **IntList** class. You may **not** use any *Standard Template Library* containers (such as **`list`** or **`forward_list`**)..

Also create a **Lab18a.cpp** source code file. This file contains the "main()" function.

Class Specification File (IntList.h)

In the **IntList.h** file, declare a class named **`IntList`**. The class must declare a **`struct`** for a linked list of integers:

```
struct ListNode
{
     int value;
     struct ListNode *next;
}
```

The **`IntList`** class must contain a member variable that is a pointer to ListNode:

```
ListNode *head;     //  List head pointer
```

The **`IntList`** class must define a constructor that takes no arguments and sets the **`head`** member variable to the **`nullptr`** pointer value.

The **`IntList`** class must also declare function prototypes for the following member functions:

```
~IntList();
void appendNode(int);      // Append a new node at the end of the list.
void insertNode(int);      // Insert a new node into the list, maintaining a sorted order.
void deleteNode(int);      // remove a node from the list.
void print();              // output the contents of the list to the screen
int length();              // display the list length (number of nodes) to the caller.
void maxValue();           // display the largest data value in the list.
int total();               // return the total (sum) of all list values to the caller.
```

Class Implementation File (IntList.cpp)

The **`IntList.cpp`** file must contain code for all of the functions declared in **`IntList.h`**.

Interactive "Main" Program

In the main function, implement an interactive command-loop similar to those we have used in other labs. (Feel free to copy portions of code from **your** solution to an earlier lab exercise, if you wish.)

Interactive Commands

The program must support the following commands:

- **a** APPEND a new node at the end of the list
- **c** COUNT the nodes in the list (display the list length).
- **d** DELETE a node from the list
- **i** INSERT a node into the list, maintaining the sorted order.
- **m** display the MAXIMUM (largest) value in the list.
- **p** PRINT the contents of the list on the console. Format the output as shown in the **Sample Output** section of this document. That is, the output must include the memory address of each node, the value field, and the value of the next pointer. (We should be able to look at the console output and follow the links from one node to the next in the list.)
- **h** HELP text
- **q** QUIT (end the program)
- **t** display the TOTAL (sum) of all values in the list.

The APPEND, DELETE, and INSERT commands must prompt the user to enter an integer value.

If the list is empty, the MAXIMUM command must display a "List is empty" message.

Sample Output

In the sample session shown (below and on the following pages), the text that the user types is shown in **`bold`** font. In actuality, all text appears in the same font. (As always, the exact memory addresses of dynamically allocated memory are potentially different each time the program runs.)

Sample Input / Output

```
Command: h
Supported commands:
        a     APPEND a value to the list.
        c     COUNT the nodes in the list (display LENGTH of the list).
        d     DELETE a value from the list.
        i     INSERT a value into the list.
        m     display MAXIMUM (largest) value in the list.
        p     PRINT the list contents.
        h     print this help text.
        q     quit (end the program).
        t     display TOTAL (sum) of all values in the list.
```

Sample Input / Output

```
Command: m
List is empty.

Command: t
Total of all list values = 0.

Command: i
Enter number to insert into the list: 12

Command: i
Enter number to insert into the list: 4

Command: i
Enter number to insert into the list: 32

Command: p
head=000001C373181CB0
000001C373181CB0:  value=   4 next=000001C373181990
000001C373181990:  value=  12 next=000001C373181F30
000001C373181F30:  value=  32 next=0000000000000000

Command: a
Enter number to append to the list: 5

Command: p
head=000001C373181CB0
000001C373181CB0:  value=   4 next=000001C373181990
000001C373181990:  value=  12 next=000001C373181F30
000001C373181F30:  value=  32 next=000001C3731813F0
000001C3731813F0:  value=   5 next=0000000000000000

Command: m
Maximum value in list = 32

Command: t
Total of all list values = 53.

Command: d
Enter number to delete from the list: 5

Command: t
Total of all list values = 48.

Command: m
Maximum value in list = 32

Command: p
```

Sample Input / Output

```
head=000001C373181CB0
000001C373181CB0:  value=   4 next=000001C373181990
000001C373181990:  value=  12 next=000001C373181F30
000001C373181F30:  value=  32 next=0000000000000000

Command: i
Enter number to insert into the list: 23

Command: i
Enter number to insert into the list: 9

Command: c
Length of list = 5 nodes

Command: p
head=000001C373181CB0
000001C373181CB0:  value=   4 next=000001C373181530
000001C373181530:  value=   9 next=000001C373181990
000001C373181990:  value=  12 next=000001C3731816C0
000001C3731816C0:  value=  23 next=000001C373181F30
000001C373181F30:  value=  32 next=0000000000000000

Command: i
Enter number to insert into the list: 2

Command: p
head=000001C373181FD0
000001C373181FD0:  value=   2 next=000001C373181CB0
000001C373181CB0:  value=   4 next=000001C373181530
000001C373181530:  value=   9 next=000001C373181990
000001C373181990:  value=  12 next=000001C3731816C0
000001C3731816C0:  value=  23 next=000001C373181F30
000001C373181F30:  value=  32 next=0000000000000000

Command: i
Enter number to insert into the list: 65

Command: p
head=000001C373181FD0
000001C373181FD0:  value=   2 next=000001C373181CB0
000001C373181CB0:  value=   4 next=000001C373181530
000001C373181530:  value=   9 next=000001C373181990
000001C373181990:  value=  12 next=000001C3731816C0
000001C3731816C0:  value=  23 next=000001C373181F30
000001C373181F30:  value=  32 next=000001C373181850
000001C373181850:  value=  65 next=0000000000000000

Command: c
Length of list = 7 nodes
```

Sample Input / Output

```
Command: m
Maximum value in list = 65

Command: t
Total of all list values = 147.

Command: d
Enter number to delete from the list: 12

Command: p
head=000001C373181FD0
000001C373181FD0:  value=   2 next=000001C373181CB0
000001C373181CB0:  value=   4 next=000001C373181530
000001C373181530:  value=   9 next=000001C3731816C0
000001C3731816C0:  value=  23 next=000001C373181F30
000001C373181F30:  value=  32 next=000001C373181850
000001C373181850:  value=  65 next=0000000000000000

Command: c
Length of list = 6 nodes

Command: d
Enter number to delete from the list: 8
Data value 8 not found.

Command: p
head=000001C373181FD0
000001C373181FD0:  value=   2 next=000001C373181CB0
000001C373181CB0:  value=   4 next=000001C373181530
000001C373181530:  value=   9 next=000001C3731816C0
000001C3731816C0:  value=  23 next=000001C373181F30
000001C373181F30:  value=  32 next=000001C373181850
000001C373181850:  value=  65 next=0000000000000000

Command: d
Enter number to delete from the list: 2

Command: p
head=000001C373181CB0
000001C373181CB0:  value=   4 next=000001C373181530
000001C373181530:  value=   9 next=000001C3731816C0
000001C3731816C0:  value=  23 next=000001C373181F30
000001C373181F30:  value=  32 next=000001C373181850
000001C373181850:  value=  65 next=0000000000000000

Command: d
Enter number to delete from the list: 65

Command: p
```

Sample Input / Output

```
head=000001C373181CB0
000001C373181CB0:  value=   4 next=000001C373181530
000001C373181530:  value=   9 next=000001C3731816C0
000001C3731816C0:  value=  23 next=000001C373181F30
000001C373181F30:  value=  32 next=0000000000000000

Command: c
Length of list = 4 nodes

Command: m
Maximum value in list = 32

Command: t
Total of all list values = 68.

Command: d
Enter number to delete from the list: 32

Command: p
head=000001C373181CB0
000001C373181CB0:  value=   4 next=000001C373181530
000001C373181530:  value=   9 next=000001C3731816C0
000001C3731816C0:  value=  23 next=0000000000000000

Command: m
Maximum value in list = 23

Command: t
Total of all list values = 36.

Command: d
Enter number to delete from the list: 4

Command: p
head=000001C373181530
000001C373181530:  value=   9 next=000001C3731816C0
000001C3731816C0:  value=  23 next=0000000000000000

Command: c
Length of list = 2 nodes

Command: t
Total of all list values = 32.

Command: m
Maximum value in list = 23

Command: d
Enter number to delete from the list: 9
```

Sample Input / Output

```
Command: c
Length of list = 1 nodes

Command: p
head=000001C3731816C0
000001C3731816C0:  value=  23 next=0000000000000000

Command: d
Enter number to delete from the list: 23

Command: c
Length of list = 0 nodes

Command: m
List is empty.

Command: q
Exiting program with status = 0
```

If your program is working correctly, then **your** results should be the same as shown in the **Sample Input/Output** above.

Submit and Demonstrate the Working Program

As ***always***, you must **submit** the source code file(s) and **demonstrate** the solution during class, in order to get credit for completing this assignment.

- Submit the source code files (**`*.cpp, *.h`**) for the working program to the *Moodle* assignment for this Lab Exercise.
- **Demonstrate** the working program to the instructor during class.
- Be sure to save a copy of the source files (**`*.cpp, *.h`**) in a safe place for future reference.

> **NOTE**: when you view the contents of a directory (folder) on your computer, the "**`.cpp`**" or "**`.h`**" filename suffix may not be visible. However, most systems display some sort of "Type" information for each file.