1. Download the following code and text from the website and create a workspace:

<https://markbowman.org/231/Lab13.zip>

2. Compile and run the program. Use Names.txt as the input file. You should see a listing of the file.

3. Replace the shaded code with this:

p = new node(name);

p->put(cout);

delete p;

4. Run the program again. You should see a listing of the nodes in triplicate.

5. What do each of the three copies of the node represent?

6. Add the following code to the beginning of the function, before the file is opened:

head = NULL;

7. Add this code at the end of the program, after the file is closed:

if(head!=NULL) delete head;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[2. Compile and run the program. Use Names.txt as the input file. You should see a listing of the file.](https://markbowman.org/231/Lab13.zip)

[3. Replace the shaded code with this:](https://markbowman.org/231/Lab13.zip)

[p = new node(name);](https://markbowman.org/231/Lab13.zip)

[p->put(cout);](https://markbowman.org/231/Lab13.zip)

[delete p;](https://markbowman.org/231/Lab13.zip)

[4. Run the program again. You should see a listing of the nodes in triplicate.](https://markbowman.org/231/Lab13.zip)

[5. What do each of the three copies of the node represent?](https://markbowman.org/231/Lab13.zip)

[6. Add the following code to the beginning of the function, before the file is opened:](https://markbowman.org/231/Lab13.zip)

[head = NULL;](https://markbowman.org/231/Lab13.zip)

[7. Add this code at the end of the program, after the file is closed:](https://markbowman.org/231/Lab13.zip)

[if(head!=NULL) delete head;](https://markbowman.org/231/Lab13.zip)

\* main()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void main()

{ string name;

fstream infile;

node \*head,\*p,\*q;

// Open file

cout << "Enter file name: ";

cin >> name;

cout << left << endl;

infile.open(name,ios::in);

// Loop through file

while(!infile.eof())

{ infile >> name;

// Process valid input

if(infile.good())

{ cout << name << endl;

};

};

// Close file

infile.close();

cout << endl;

}

8. Replace the shaded code again (remove the code from step 3):

A close up of a logo

Description automatically generated p = new node(name);

if(head==NULL)

head = p;

else

q->add(p);

q = p;

The first time a node is created, p is assigned to head. Each time after that, the node at p is added to the node that q is pointing to. Each time around the loop, q is updated to point to the end of the list.

9. Compile and run your program. You should see the messages from the constructor and destructor functions.

Text

Description automatically generated

10. Create a while loop at the end of the main() function, after the file is closed, but before the nodes are deleted, that will use p to traverse the linked list and display the contents of each node. The main() function is declared a friend to node so you can access the \*next pointer in that part of your program.

p = head;

while(p!=NULL)

{ p->put(cout);

p = p->next;

};

11. Compare your output to the sample below. Your program will use different addresses, but you should be able to see the pattern that the next pointer for each node matches the this address for the following one in the list.

Enter file name: ***Names.txt***

00670D60 Constructed

00670B30 Constructed

006709A0 Constructed

00670BD0 Constructed

00671030 Constructed

00670EA0 Constructed

00670DB0 Constructed

00670E00 Constructed

00670D10 Constructed

00670D60 Carlos 00670B30

00670B30 Ann 006709A0

Richard’s next

006709A0 Richard 00670BD0

David’s this

00670BD0 David 00671030

00671030 Mark 00670EA0

00670EA0 Frank 00670DB0

00670DB0 Susan 00670E00

00670E00 Kathy 00670D10

John’s next is NULL

00670D10 John 00000000

00670D60 Destructed

00670B30 Destructed

006709A0 Destructed

00670BD0 Destructed

00671030 Destructed

00670EA0 Destructed

00670DB0 Destructed

00670E00 Destructed

00670D10 Destructed

12. Compile and run the program with Names.txt and Numbers.txt, and save the outputs.

**Names.txt - Output:**

Text

Description automatically generated

**Numbers.txt – Output:**

Text

Description automatically generated

Response – Question 5:

The nodes printed in triplicate represent:

* First, the constructor which sets the head and tail of the pointers to null first – the list starts off empty before it reads in the list of names/numbers from file.
* Second, after the hexadecimal number is the item in the list which is printed (in the case of Names.txt, the names are Ana, Richard, etc.). That means that entry has been read but instead of a pointer to the next node there is NULL.
* Then after the entry has been read in successfully, the node for that entry is deconstructed. Then, the node for the next entry is constructed, starts off as null for the head and tail, reads in the name from the list, and repeats the second bullet above.

Main.cpp:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Lab 13

\* Created by Safford, Twymun

\* Date: 06-Oct-2021

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

#include "Node.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* main()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void main()

{

string name;

fstream infile;

node\* head, \* p, \* q;

//initialize null

head = NULL;

// Open file

cout << "Enter file name: ";

cin >> name;

cout << left << endl;

infile.open(name, ios::in);

// Loop through file

while (!infile.eof())

{

infile >> name;

// Process valid input

if (infile.good())

{

//cout << name << endl;

//replacement 1

//p = new node(name);

//p->put(cout);

//delete p;

//replacement 2

p = new node(name);

if (head == NULL)

{

head = p;

}

else

{

q->add(p);

}

q = p;

};

};

// Close file

infile.close();

cout << endl;

// Display

p = head;

while (p != NULL)

{

p->put(cout);

p = p->next;

};

cout << endl;

//clean-up - delete the head after no longer needed

if (head != NULL)

{

delete head;

}

}