

# Bonus Slides

The slides below are extras from a special class presentation.

Note: These slides will be here only for a few days.

## Implementing and Testing Programming Assignment 3

### Implementing and Testing Program 3

**Programming Assignment 3**

*Your dates appear here.*

**Statement of Work**  
**Programming Assignment 3**

**1.0 Overview**  
Madame Paster, the head librarian at Hesperia has expressed an interest in the new "Computational Maps" in Professor Dunderhead's and has asked if it would be possible to stress a program which would catalog and store information on all of the books in the Hesperia Library. Professor Dunderhead has requested just such a utility program which will be implemented as a binary tree.

**1.0 Requirements**  
The student shall define, develop, document, prototype, test, and modify as required the software system.

**1.1 This software system shall define a class, called Library which maintains a Binary Tree of instances of a structure called Book as defined in section 2.1.**

**1.2 The Library class shall implement all the functionality of a binary tree and shall contain variables and functions as described below:**

**2.1.1 Member variables:** this class shall contain one private member variable. This shall be a pointer to a Book structure and shall be named `as_ptrRoot`. The variable `as_ptrRoot` shall point to the first instance of Book in the tree.

**2.1.2 Library:** Library: these functions are the default constructor and destructor. The destructor function shall delete all instances of Book from the tree by calling the recursive function destroyTree passing as the root.

**2.1.3 findBookId:** findBookId: "findBookId" this function takes a single argument: a character array giving the name of a data file (provided by the instructor). It will open the data file and read all data from the data file, create Book structures to hold the data, and call the addBook() function to add each book to the database. This function shall be provided by the instructor. This function shall be public.

**2.1.4 findAddBookId:** "findAddBookId" this function takes a single argument: a pointer to a Book structure. It will then add this book to the binary tree of books. This function shall use the book number of the book as the key for insertion into the binary tree. It will return true if the book was successfully added to the tree or false if it failed to add the book. This function shall be public.

**2.1.5 Book "removeBookId":** "removeBookId" this function takes a book identification number as its only argument. It shall locate the book in the tree of books and remove the book from the binary tree. It will set the left and right pointers of the removed Book structure to NULL, and return a pointer to the book. If the tree was not found in the tree it shall return NULL. This function shall be public.

**Remember that if the node to be removed from the tree has two children it actually gets a reservation. In this case you will need to create a duplicate Book structure, and copy the data out of the one to be overwritten. Then after completing the deletion for a node with two children return the pointer to the new duplicate Book.**

**2.1.6 Book "getBookId":** "getBookId" this function takes a book identification number as its only argument. It will then locate the book in the tree of books and return a pointer to the book. If the book was not found it shall return NULL. This may return, as the case, that it is OK, to return a pointer to a book in the tree without worrying about a going across to other areas in the tree. This function shall be public.

**2.1.7 Book "getBookId":** "getBookId" this function takes a book title as a character array as its only argument. It just calls the private recursive getBookId() function and returns whatever that call returns. This function shall be public.

**2.1.8 Book "getBookId":** "getBookId" this function takes a book title as a character array, and a pointer to the root of a sub-tree as its only argument. It will then locate the book in the tree of books and return a pointer to the book. If the book was not found it shall return NULL. This may be a variation on the recursive two-root of a binary tree algorithm. (Note: It will take some careful thought to be sure this one works correctly. We will discuss this function in class.) This may return, as the case, that it is OK, to return a pointer to a book in the tree without worrying about a going across to other areas in the tree. This function shall be private.

**2.1.9 void printLibrary:** this is a public function which will just call the private function printLibrary() as defined in section 2.2.10.

**2.2.10 void printLibrary:** "printLibrary" this function shall take a pointer to a Book structure and print on the screen all information in the structure. This shall include the book identification number, book title, and author. This function shall be private as it should only be called by the printLibrary() function defined in section 2.2.10.

**2.2.11 void printLibrary:** "printLibrary" this shall be a private function which is called by the public function printLibrary(). It takes a single argument: a pointer to the root of a tree. It will perform an in-order recursive traversal of the tree and print all information in each book in the tree. It can call the function printBook() to print the information on the Book passed as its only argument.

**2.2.12 void getBookId:** "getBookId" this shall be a private function which can be called by the findBookId() function. It will read the next data line from the file referenced by the self argument and place it in the character array passed to it by the caller argument. This function will be provided by the instructor.

**2.2.13 void destroyBook:** "destroyBook" this shall be a private function which can be called by the class destructor. It will recursively traverse the tree and delete all nodes from the tree.

**2.3 This software system shall define a structure called Book which contains all information to define a book as the Hesperia library. Code defining this structure shall be stored in a header file called Book.h. A Book.h file is included in the zip file which can be downloaded from the link given below.**

**2.3.1 An int, called bookNumber:** This variable shall be used as the key.

**2.3.2 A character array, called Title:** capable of holding strings of up to 127 characters. This array shall hold the title of a book.

**2.3.3 A character array, called Author:** capable of holding strings of up to 64 characters. This array shall hold the author's name.

**2.3.4 Two pointers, called left and right:** which shall point to Book structures and allow the building of binary trees of Book structures.

**2.4 The Library class file and its associated header file must be capable of being compiled and linked as with a device program, the way any C++ program.**

**3.0 Deliverables**  
Three products shall be delivered to the instructor electronically via e-mail as specified below:

**3.1 Source Report:** The student shall provide filled out Source Report form, for instructor approval. SRT (Short Letter Three) Monday, July 25.

**3.2 Program source file:** The student shall provide fully tested electronic copies of the .cpp and .h files. These files must be submitted to the instructor via e-mail. The files shall be delivered SRT Monday, July 25.

**4.0 Period of Performance**  
The period of performance of this assignment is 75 days from the date of assignment. Files that will not compile will not be accepted. If an error is found during testing the instructor will request the error and give you a chance to correct the error and resubmit the files with no penalty as long as the files are resubmitted before the DDD.

This programming assignment will be graded by taking your class definition file (Library.cpp) and linking with a test driver program which will automatically test the Library class. Before you will find two functions which can be used in your Library class.

**2.1.8 Book "getBookId":** "getBookId" this function takes a book title as a character array, and a pointer to the root of a sub-tree as its only argument. It will then locate the book in the tree of books and return a pointer to the book. If the book was not found it shall return NULL. This may return, as the case, that it is OK, to return a pointer to a book in the tree without worrying about a going across to other areas in the tree. This function shall be private.

**2.1.9 void printLibrary:** this is a public function which will just call the private function printLibrary() as defined in section 2.2.10.

**2.2.10 void printLibrary:** "printLibrary" this function shall take a pointer to a Book structure and print on the screen all information in the structure. This shall include the book identification number, book title, and author. This function shall be private as it should only be called by the printLibrary() function defined in section 2.2.10.

**2.2.11 void printLibrary:** "printLibrary" this shall be a private function which is called by the public function printLibrary(). It takes a single argument: a pointer to the root of a tree. It will perform an in-order recursive traversal of the tree and print all information in each book in the tree. It can call the function printBook() to print the information on the Book passed as its only argument.

**2.2.12 void getBookId:** "getBookId" this shall be a private function which can be called by the findBookId() function. It will read the next data line from the file referenced by the self argument and place it in the character array passed to it by the caller argument. This function will be provided by the instructor.

**2.2.13 void destroyBook:** "destroyBook" this shall be a private function which can be called by the class destructor. It will recursively traverse the tree and delete all nodes from the tree.

**2.3 This software system shall define a structure called Book which contains all information to define a book as the Hesperia library. Code defining this structure shall be stored in a header file called Book.h. A Book.h file is included in the zip file which can be downloaded from the link given below.**

**2.3.1 An int, called bookNumber:** This variable shall be used as the key.

**2.3.2 A character array, called Title:** capable of holding strings of up to 127 characters. This array shall hold the title of a book.

**2.3.3 A character array, called Author:** capable of holding strings of up to 64 characters. This array shall hold the author's name.

**2.3.4 Two pointers, called left and right:** which shall point to Book structures and allow the building of binary trees of Book structures.

**2.4 The Library class file and its associated header file must be capable of being compiled and linked as with a device program, the way any C++ program.**

**3.0 Deliverables**  
Three products shall be delivered to the instructor electronically via e-mail as specified below:

**3.1 Source Report:** The student shall provide filled out Source Report form, for instructor approval. SRT (Short Letter Three) Monday, July 25.

**3.2 Program source file:** The student shall provide fully tested electronic copies of the .cpp and .h files. These files must be submitted to the instructor via e-mail. The files shall be delivered SRT Monday, July 25.

**4.0 Period of Performance**  
The period of performance of this assignment is 75 days from the date of assignment. Files that will not compile will not be accepted. If an error is found during testing the instructor will request the error and give you a chance to correct the error and resubmit the files with no penalty as long as the files are resubmitted before the DDD.

This programming assignment will be graded by taking your class definition file (Library.cpp) and linking with a test driver program which will automatically test the Library class. Before you will find two functions which can be used in your Library class.

*These slides can be found at <http://www.cs.uah.edu/~rcoleman/CS221/Temp/HWP3Hints.html>*

Testing Program 3 \_1

# Implementing and Testing Program 3

## Required Files\*

Source file (.cpp) **Library.cpp**  
Header file (.h) **Library.h**  
Header file (.h) **Book.h**

## Library Private Member Variable\*

(1) Pointer to an instance of Book called **m\_pRoot**

## Possessions Class Methods\*

### Public Functions

- (1) **Library()** - Default constructor. Sets m\_pRoot to NULL.
- (2) **~Library()** - Destructor. Calls destroyTree passing in m\_pRoot to delete all instances of Book in the tree.
- (3) **buildLibrary(char \*fileName)** - Read the data file and build the tree. Function provided by the instructor.
- (3) **bool addBook(Book \*newBook)** - Add a Book to the binary tree of Books.
- (4) **Book \*removeBook(int bookNum)** - Find an item in the tree, remove and return it.
- (5) **Book \*getBookByNumber(int bookNum)** - Locate a Book in the tree searching by its' ID number and return a pointer to it or NULL if not found.
- (6) **Book \*getBookByTitle(char \*title)** - Call the private getBookByTitle and return what it returns.
- (7) **void printLibrary()** - Function to initiate printing of all books in the tree.

### Private Functions

- (1) **Book \*getBookByTitle(char \*title, Book \*rt)** - Traverse the tree and locate a Book in the tree searching by its' title and return a pointer to it or NULL if not found.
- (2) **void printOne(Book \*book)** - Print all information in a single instance of Book.
- (3) **void printAll(Book \*rt)** - Traverse the tree, in order, and print all information on all Books.
- (8) **void destroyTree(Book \*rt)** - Function called by the destructor to recursively destroy each book in the tree.
- (9) **bool getNextLine(ifstream &inFile, char \*line, int lineLen)** - Read a line from the data file. Provided by the instructor.

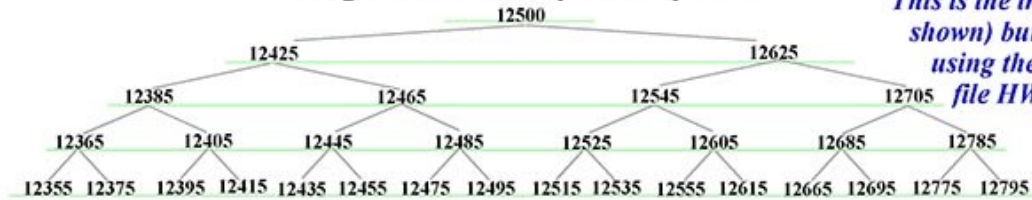
\*Spelling must be exactly as specified.

Testing Program 3\_2

# Implementing and Testing Program 3

## Hogwarts Library Binary Tree

*This is the tree (book numbers shown) built by buildTree() using the provided data file HWBookDB.txt*



*The provided data file HWBookDB.txt*

```

# Hogwarts Library Books for Programming Assignment 3
#
# The books listed below are all those mentioned in Harry Potter books
# by J.K. Rowling. Many of the authors were also created by Ms. Rowling
# others were made up by Dr. Coleman. Any resemblance to real or
# imagined people is purely a coincidence.
#
# Note: the books are listed by their book number in an order that will
# ensure that a binary tree using the book number as the key will
# be as close to balanced as possible.
#
12500
The Standard Book of Spells
Miranda Goshawk
#
12425
A History of Magic
Bathilda Bagshot
#
12025
Magical Theory
Adalbert Waffling
#
12385
A Beginner's Guide to Transfiguration
Emelic Switch
#
12405
One Thousand Magical Herbs and Fungi
Phyllida Spore
#
12545
Magical Drafts and Potions
Arsenius Jigger
#
12705
Fantastic Beasts and Where to Find Them
Newt Scamander
#
12365
The Dark Forces: A Guide to Self-Protection
Quentin Trimble
#
12405
Curses and Counterurses
Vindictus Viridian
#
12445
Great Wizards of the Twentieth Century
Horatio Twilbee
#
12485
Quidditch Through the Ages
Kennilworthy Whisp
#
12525
Dragon Breeding for Pleasure and Profit
Draco Wymn
#
12805
Break with a Banshee
Gilderoy Lockhart
#
12685
Gedding with Ghouls
Gilderoy Lockhart
#
12785
Travels with Troils
Gilderoy Lockhart
#
12355
Voyages with Vampires
Gilderoy Lockhart

```

```

#
12375
Wanderings with WereWolves
Gilderoy Lockhart
#
12395
Year with the Yeti
Gilderoy Lockhart
#
12415
Magical Me
Gilderoy Lockhart
#
12435
Most Potent Potions
Chelone Glabra
#
12455
Protects Who Gained Power
Maria Leach
#
12475
Flying with the Canons
Jason Calbre
#
12495
The Monster Book of Monsters
Krecher Krowley
#
12515
The Invisible Book of Invisibility
Veronica Void
#
12535
Unfogging the Future
Cassandra Vablatsky
#
12555
Broken Balls: When Fortunes Turn Foul
Cassandra Vablatsky
#
12615
Death Omens: What to Do When You Know the Worst is Coming
Cassandra Vablatsky
#
12665
Hogwarts, A History
Armando Dippet
#
12695
The Life and Lies of Albus Dumbledore
Rita Skeeter
#
12775
The Tales of Beedle the Bard
Gladys Patron
#
12795
Armando Dippet: Master or Moron
Rita Skeeter

```

Testing Program 3\_3

# Implementing and Testing Program 3

*This only covers the final testing of the completed application.  
It does not include the testing required at the end of Sprint 1.*

## Testing the addBook function:

```
bool addBook(Book *newBook)
```

## Testing the print functions:

```
void printLibrary()
```

```
void printOne(Book *book)
```

```
void printAll(Book *rt)
```

```
Library *lib = new Library();  
lib->buildLibrary("HWBookDB.txt");  
lib->printLibrary();  
// Check that all books are listed and in order by  
// book number
```

1. Create an instance of Library in main.
2. Using the provided buildLibrary function and the provided data file build the Library.
3. After adding all the books call the public function printLibrary and check that all books have been added and that they are listed in order by book number.

*You cannot directly test the private functions printAll and printOne, but you can indirectly test them by call the public printLibrary function which in turn calls the private functions.*



*Remember it is your  
responsibility to test  
your program. It is  
not the instructor's  
responsibility.*

*\*Remember to comment out all the debug cout statements BEFORE turning in your files.* Testing Program 3\_4



# Implementing and Testing Program 3

*This only covers the final testing of the completed application.  
It does not include the testing required at the end of Sprint 1.*

## Testing the getBookByNumber function:

**Book \*getBookByNumber(int bookNum)**

```
Book *bk;
bk = lib->getBookByNumber(12485);
if((bk != NULL) && (strcmp(bk->Title,
    "Quidditch Through the Ages") == 0))
{
    cout << "Search successful" << endl;
// Repeat for other books and a book number not
// in the library.
```

1. Create a pointer to struct Book in main.
2. Call getBookByNumber passing in numbers of books in the Library. Check to be sure the returned pointer is not NULL and that it points to the correct book. (Repeat for the root, leaf, and interior node.)
3. Call the getBookByNumber function passing in a book number known to not be in the Library.

## Testing the getBookByTitle functions:

**Book \*getBookByTitle(char \*title)**

**Book \*getBookByTitle(char \*title, Book \*rt)**

```
Book *bk;
bk = lib->getBookByTitle("Quidditch Through the Ages");
if((bk != NULL) && (strcmp(bk->Title,
    "Quidditch Through the Ages") == 0))
{
    cout << "Search successful" << endl;
// Repeat for other books and a book title not
// in the library.
```

1. Create a pointer to struct Book in main.
2. Call getBookByTitle passing in titles of books in the Library. Check to be sure the returned pointer is not NULL and that it points to the correct book. (Repeat for the root, leaf, and interior node.)
3. Call the getBookByTitle function passing in a book number known to not be in the Library.

*You cannot directly test the private version of getBookByTitle, but you can indirectly test it by call the public getBookByTitle function which in turn calls the private function.*

*\*Remember to comment out all the debug cout statements BEFORE turning in your files.*

Testing Program 3\_5

# Testing Program 3

*This only covers the final testing of the completed application.  
It does not include the testing required at the end of Sprint 1.*

## Testing the removeBook function:

**Book \*removeBook(int bookNum)**

1. Create a different data file using a subset of the books in the provided file. See next slides for an example.
2. Call removeBook passing in the number of a book known to not be in the Library then check for NULL being returned.
2. Call removeBook passing in the number of a book known to be in the Library.
3. Check the pointer to the instance of Book returned that it is not NULL and it is the Book to be removed
4. Do this for several books making sure to check for deleting
  - (1) Node with no children, (2) node with one child on the left, (3) node with one child on the right, (4) node with two children. Test each of the above when the node to be deleted is the root and when it is another node in the tree.

### Test Checklist for removeBook

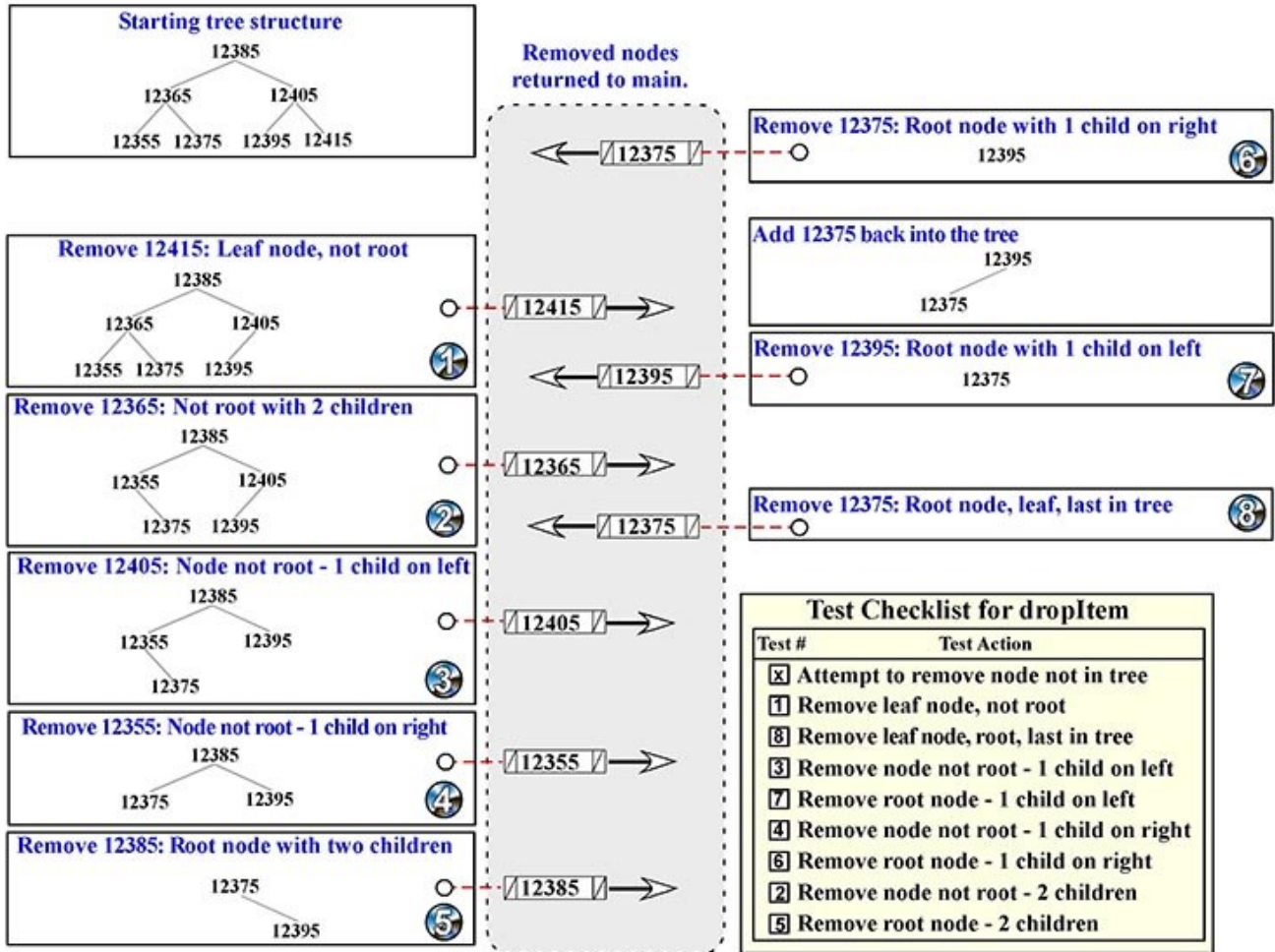
- ☐ Attempt to remove node not in tree
- ☐ Remove leaf node, not root
- ☐ Remove leaf node, root, last in tree
- ☐ Remove node not root - 1 child on left
- ☐ Remove root node - 1 child on left
- ☐ Remove node not root - 1 child on right
- ☐ Remove root node - 1 child on right
- ☐ Remove node not root - 2 children
- ☐ Remove root node - 2 children

## Sample data file for testing the removeBook Function

```
#
12385
A Beginners' Guide to Transfiguration
Emeric Switch
#
12365
The Dark Forces: A Guide to Self-Protection
Quentin Trimble
#
12405
Curses and Countercurses
Vindictus Viridian
#
12355
Voyages with Vampires
Gilderoy Lockhart
#
12375
Wanderings with WereWolves
Gilderoy Lockhart
#
12395
Year with the Yeti
Gilderoy Lockhart
#
12415
Magical Me
Gilderoy Lockhart
```

*See next slide for details of this testing.*

# Testing Program 3



Testing Program 3\_7



# Implementing and Testing Program 3

## Library.cpp - Build Library Function

```
//-----
// Function: buildLibrary()
// Purpose: Build the library from a data file
// Args: fileName - character array holding the name of the
//        datafile defining the library.
// Returns: True if library was successfully built.
//-----
bool Library::buildLibrary(char *fileName)
{
    ifstream inFile;
    Book *bk;
    char line[128];

    inFile.open(fileName, ifstream::in);
    if(!inFile.is_open())
    {
        // inFile.is_open() returns false if the file could not be found or
        // if for some other reason the open failed.
        cout << "Unable to open file " << fileName << ".\nProgram terminating...\n";
        return false;
    }

    while (getNextLine(inFile, line, 127)) // While the next line is readable
    {
        // Create a new Book and set it's pointers to NULL
        bk = new Book();
        bk->left = bk->right = NULL;

        // Set the book number
        bk->bookNumber = atoi(line);

        // Read the book title
        getNextLine(inFile, line, 127);
        strcpy(bk->Title, line);

        // Read the author
        getNextLine(inFile, line, 127);
        strcpy(bk->Author, line);

        // Add this book to the tree
        addBook(bk);
    }
    return true;
}
```

*This function will add Books to the Library by calling the addBook function which implements the insert into a binary tree algorithm*

*Note: This is also testing the addBook function.*

*This function is provided by the Instructor*

Testing Program 3\_8



# Implementing and Testing Program 3

## Library.cpp - Get Next Line Function

```
//-----
// Function: getNextLine()
// Purpose: Read a line from the data file skipping blank lines
//          and comment lines beginning with #
// Args: inFile - reference argument to an open ifstream object
//        to read from.
//        line - character array into which the data line is read
//        lineLen - maximum number of characters which can be read
//               into the array line
// Returns: True if a successful read was done. If False is
//          returned then the array line will be zero length.
//-----
bool Library::getNextLine(ifstream &inFile, char *line, int lineLen)
{
    int done = false;
    while(!done)
    {
        inFile.getline(line, lineLen);
        if(inFile.good()) // If a line was successfully read
        {
            if(strlen(line) == 0) // Skip any blank lines
                continue;
            else if(line[0] == '#') // Skip any comment lines
                continue;
            else done = true; // Got a valid data line so return with this line
        }
        else
        {
            strcpy(line, "");
            return false; // Flag end of file with null string and return false
        }
    } // end while
    return true; // Flag successful read
}
```

*This function is provided by the Instructor*

Testing Program 3\_9