```
******************************************************
*  PROGRAMMED BY : Ali Eshghi & Amirarsalan Valipour
*  CLASS         : CS1B
*  SECTION       : MW: 7:30p - 9:50p
*  LAB #10       : Creating an Ordered List
******************************************************

LIST MENU:
1 - Create List
2 - Display List
3 - Is the list empty?
4 - Search by name
5 - Remove Node
6 - Clear List
0 - to Exit
Enter a command (0 to exit): 1

Adding: Payne, Royal
Adding: Ding, Bill
Adding: Post, Mark
Adding: Sassin, Anna
Adding: Lear, Shanda
Adding: Longbottom, Iva
Adding: Dwyer, Barb
Adding: Hogg, Ima
Adding: Belcher, Ura
Adding: Age, Sue


LIST MENU:
1 - Create List
2 - Display List
3 - Is the list empty?
4 - Search by name
5 - Remove Node
6 - Clear List
0 - to Exit
Enter a command (0 to exit): 2

   #    NAME                     GENDER   AGE
------- ------------------------  -------- -----
   1    Age, Sue                    F      32
   2    Belcher, Ura                F      46
   3    Ding, Bill                  M      21
   4    Dwyer, Barb                 F      24
   5    Hogg, Ima                   F      43
   6    Lear, Shanda                F      18
   7    Longbottom, Iva             F      45
   8    Payne, Royal                M      73
   9    Post, Mark                  M      20
  10    Sassin, Anna                F      62

LIST MENU:
1 - Create List
2 - Display List
```

```
3 - Is the list empty?
4 - Search by name
5 - Remove Node
6 - Clear List
0 - to Exit
Enter a command (0 to exit): 3

No, The list is NOT empty.

LIST MENU:
1 - Create List
2 - Display List
3 - Is the list empty?
4 - Search by name
5 - Remove Node
6 - Clear List
0 - to Exit
Enter a command (0 to exit): 4

Who would you like to search for? Age, Sue

Searching for Age, Sue...

Name:    Age, Sue
Gender:  F
Age:     32

LIST MENU:
1 - Create List
2 - Display List
3 - Is the list empty?
4 - Search by name
5 - Remove Node
6 - Clear List
0 - to Exit
Enter a command (0 to exit): 4

Who would you like to search for? Sassin, Anna

Searching for Sassin, Anna...

Name:    Sassin, Anna
Gender:  F
Age:     62

LIST MENU:
1 - Create List
2 - Display List
3 - Is the list empty?
4 - Search by name
5 - Remove Node
6 - Clear List
0 - to Exit
Enter a command (0 to exit): 4

```

```
Who would you like to search for? Ding, Bill

Searching for Ding, Bill...

Name:    Ding, Bill
Gender:  M
Age:     21

LIST MENU:
1 - Create List
2 - Display List
3 - Is the list empty?
4 - Search by name
5 - Remove Node
6 - Clear List
0 - to Exit
Enter a command (0 to exit): 4

Who would you like to search for? Smith, Will

Searching for Smith, Will...

I'm sorry, "Smith, Will" was NOT found!

LIST MENU:
1 - Create List
2 - Display List
3 - Is the list empty?
4 - Search by name
5 - Remove Node
6 - Clear List
0 - to Exit
Enter a command (0 to exit): 5

Who would you like to remove?  Age, Sue

Removing Age, Sue!

LIST MENU:
1 - Create List
2 - Display List
3 - Is the list empty?
4 - Search by name
5 - Remove Node
6 - Clear List
0 - to Exit
Enter a command (0 to exit): 5

Who would you like to remove?  Post, Mark

Removing Post, Mark!

LIST MENU:
1 - Create List
2 - Display List
```

```
3 - Is the list empty?
4 - Search by name
5 - Remove Node
6 - Clear List
0 - to Exit
Enter a command (0 to exit): 5

Who would you like to remove?  Sassin, Anna

Removing Sassin, Anna!

LIST MENU:
1 - Create List
2 - Display List
3 - Is the list empty?
4 - Search by name
5 - Remove Node
6 - Clear List
0 - to Exit
Enter a command (0 to exit): 5

Who would you like to remove?  Smith, Will

I'm sorry, "Smith, Will" was NOT found!

LIST MENU:
1 - Create List
2 - Display List
3 - Is the list empty?
4 - Search by name
5 - Remove Node
6 - Clear List
0 - to Exit
Enter a command (0 to exit): 6

CLEARING LIST:
Removing Belcher, Ura
Removing Ding, Bill
Removing Dwyer, Barb
Removing Hogg, Ima
Removing Lear, Shanda
Removing Longbottom, Iva
Removing Payne, Royal

LIST MENU:
1 - Create List
2 - Display List
3 - Is the list empty?
4 - Search by name
5 - Remove Node
6 - Clear List
0 - to Exit
Enter a command (0 to exit): 2

Can't Display an empty list!
```

```

LIST MENU:
1 – Create List
2 – Display List
3 - Is the list empty?
4 - Search by name
5 - Remove Node
6 - Clear List
0 – to Exit
Enter a command (0 to exit): 3

Yes, the list is empty.

LIST MENU:
1 – Create List
2 – Display List
3 - Is the list empty?
4 - Search by name
5 - Remove Node
6 - Clear List
0 – to Exit
Enter a command (0 to exit): 4

Search operation is not available for an empty list.

LIST MENU:
1 – Create List
2 – Display List
3 - Is the list empty?
4 - Search by name
5 - Remove Node
6 - Clear List
0 – to Exit
Enter a command (0 to exit): 5

Remove operation is not available for an empty list.

LIST MENU:
1 – Create List
2 – Display List
3 - Is the list empty?
4 - Search by name
5 - Remove Node
6 - Clear List
0 – to Exit
Enter a command (0 to exit): 6

The list has been cleared!

LIST MENU:
1 – Create List
2 – Display List
3 - Is the list empty?
4 - Search by name
5 - Remove Node
```

```
6 - Clear List
0 - to Exit
Enter a command (0 to exit): x

**** Please input a NUMBER between 0 and 6 ****

LIST MENU:
1 - Create List
2 - Display List
3 - Is the list empty?
4 - Search by name
5 - Remove Node
6 - Clear List
0 - to Exit
Enter a command (0 to exit): 7

**** The number 7 is an invalid entry     ****
**** Please input a number between 0 and 6 ****

LIST MENU:
1 - Create List
2 - Display List
3 - Is the list empty?
4 - Search by name
5 - Remove Node
6 - Clear List
0 - to Exit
Enter a command (0 to exit): -1

**** The number -1 is an invalid entry     ****
**** Please input a number between 0 and 6 ****

LIST MENU:
1 - Create List
2 - Display List
3 - Is the list empty?
4 - Search by name
5 - Remove Node
6 - Clear List
0 - to Exit
Enter a command (0 to exit): 0

```

```
/*************************************************************************
* AUTHOR     : Amirarsalan Valipour & Ali Eshghi
* STUDENT ID : 1103126 - 1112261
* LAB #10    : Creating an Ordered List
* CLASS      : CS 1B
* SECTION    : MW - 7:30 pm
* DUE DATE   : 11/05/2019
*************************************************************************
* PrintMenu
*************************************************************************
*  This function will output the menu, gets the user's option and checks to see
*    if the input is eligible or not.
*************************************************************************/

#ifndef MYHEADER_H_
#define MYHEADER_H_

#include<iostream>
#include<iomanip>
#include<string>
#include<fstream>
#include<limits>
#include<sstream>
using namespace std;

const string NAME = "Creating an Ordered List";
const char   TYPE    = 'L';
const int    NUM     = 10 ;
const string CLASS   = "CS1B";
const string SECTION = "MW: 7:30p - 9:50p";



enum MenuOption
{
    EXIT,
    CREATE,
    DISPLAY,
    ISEMPTY,
    SEARCH,
    REMOVE,
    CLEAR
};

struct PersonNode
{
    string     name;
    char   gender;
    int        age;
    PersonNode *next;
    PersonNode *prev;
};

/*************************************************************************
 * Function - PrintHeaderFile
 * --------------------------------------------------------------------------
 * This function will output the class heading to the screen.
 *
 * return type - nothing
 *               the function is void type
 ****************************************************************************/
void PrintHeaderFile();

/*****************************************************************************
 * Function - PrintMenu
 * --------------------------------------------------------------------------
 * This function will output menu option to the screen and waits for the user to
 * input an option to what to what to do. The options are as following:
 *
 * 1 - Create List
 * 2 - Display List
 * 3 - Is the list empty?
 * 4 - Search by name
 * 5 - Remove Node
 * 6 - Clear List
 * 0 - to Exit
 *
 * return type - Integer
 ****************************************************************************/
int PrintMenu();

/*****************************************************************************
 * Function - CreatList
 * --------------------------------------------------------------------------
 * This function will get the data from the input file and puts the data into
 * the contents of the nodes, then add the nodes to the empty lists passed by
 * Reference to the function
 *
 * return type - nothing
 *               the function is void type
 ****************************************************************************/
void CreatList(PersonNode *&head);

/*****************************************************************************
 * Function - DisplayList
 * --------------------------------------------------------------------------
 * This function will output the contents of the nodes of the linked list
 * created in the CreatList function
 *
 * return type - nothing
 *               the function is void type
 ****************************************************************************/
void DisplayList(PersonNode *head);

/*****************************************************************************
 * Function - IsEmpty
 * --------------------------------------------------------------------------
 * This function check if the list created or modified by the user is empty or
 * no and then  outputs if the list is empty or no
 *
 * return type - nothing
 *               the function is void type
 ******************************************************************************/
void IsEmpty(PersonNode *head);

/******************************************************************************
 * Function - SearchName
 * ----------------------------------------------------------------------------
 * This function will get a name from the user and search the name content of
 * each node to see if there is a matching name in the nodes with the name
 * searched by the user. if found, the function prints out every content of the
 * node for the user, if not found, the function outputs that the name searched
 * by the user was not found.
 *
 * return type - nothing
 *               the function is void type
 ******************************************************************************/
void SearchName(PersonNode *head);

/******************************************************************************
 * Function - RemoveNode
 * ----------------------------------------------------------------------------
 * This function will ask the user which node the user wants to remove and it
 * searches the nodes based on the name content of the nodes and if the name
 * content of a node matches the name input by the user, the function removes
 * the node from the list, if not, the function outputs that the name searched
 * by the user has not found in the list.
 *
 * return type - nothing
 *               the function is void type
 ******************************************************************************/
void RemoveNode(PersonNode *&head);

/******************************************************************************
* Function - ClearList
*****************************************************************************
*  This function Allows the user to delete all the nodes in the linked list and
*  make an empty list out of the list of the names that we had
*
*  return type - nothing
*               the function is void type
******************************************************************************/
void ClearList(PersonNode *&head);


#endif /* MYHEADER_H_ */

```

```cpp
/******************************************************************************
* AUTHOR     : Amirarsalan Valipour & Ali Eshghi
* STUDENT ID : 1103126 - 1112261
* LAB #10    : Creating an Ordered List
* CLASS      : CS 1B
* SECTION    : MW - 7:30 pm
* DUE DATE   : 11/05/2019
*******************************************************************************/

#include "MyHeader.h"


/*******************************************************************************
 * Function : PrintHeaderFile
 * ---------------------------------------------------------------------------
 * This function will output the class heading to the screen.
 ******************************************************************************/

void PrintHeaderFile()
{

    /***************************************************************************
     * OUTPUT - outputs class heading to file and then console.
     **************************************************************************/

    cout << left;
    cout << "******************************************************\n"     ;
    cout << "*  PROGRAMMED BY : Ali Eshghi & Amirarsalan Valipour      "   ;
    cout << "\n*  "     << setw(14) << "CLASS"     << ": " << CLASS         ;
    cout << "\n*  "     << setw(14) << "SECTION"    << ": " << SECTION      ;
    cout << "\n*  LAB #"<< setw(9)  << NUM     << ": " << NAME       ;
    cout << "\n*******************************************************\n\n" ;
    cout << right;

}


```

```
/**************************************************************************
* AUTHOR     : Amirarsalan Valipour & Ali Eshghi
* STUDENT ID : 1103126 - 1112261
* LAB #10    : Creating an Ordered List
* CLASS      : CS 1B
* SECTION    : MW - 7:30 pm
* DUE DATE   : 11/05/2019
**************************************************************************/


#include "MyHeader.h"

/**************************************************************************
* LAB 10
* ------------------------------------------------------------------------
* This program gets an integer as an option from the user based in the printed
* menu. the first option gets the data from an input file and put the data into
* the content of nodes and add them to an empty list that has been created in
* main. the second option displays the list that has been created in the
* first option of the menu on the screen. the second option runs a function that
* determines that the list created or modified by the user is empty or no, and
* it outputs on the screen if the list is empty or no. the fourth option let the
* user to search the nodes based on the name content of the nodes and if the
* name content of a node matches the name content of the searched name by the
* user, it shows every content of the node.the fifth option will ask the user
* which node the user wants to remove and it searches the nodes based on the
* name content of the nodes and if the name content of a node matches the name
* input by the user, the function removes the node from the list, if not, the
* function outputs that the name searched by the user has not found in the list.
* the sixth option This function Allows the user to delete all the nodes in the
* linked list and make an empty list out of the list of the names that we had.
* ------------------------------------------------------------------------
* INPUT  : option      -> menu option
* ------------------------------------------------------------------------
* PROCESS: creating the list
*          Displaying the list
*          Check if the list is empty
*          Searching by the name
*          searching and removing a name
*          clearing the list
*
*
* ------------------------------------------------------------------------
* OUTPUT : Content of the nodes
*          Information about the list based on the creating and modifying
**************************************************************************/


int main()
{
    /*************
     * VARIABLES *
     ************/

    int  menuOption;  // IN - user input
    PersonNode *head; // PROCESS - pointer for an empty list

    //creating an empty list
    head = NULL;



    //This function outputs the class heading
    PrintHeaderFile();


    // GETS USER INPUT AND CHECK IT

    //This function will output menu option to the screen and waits for the user
    //to input an option to what to what to do.
    menuOption = PrintMenu();

    while (menuOption != 0)
    {
        if (menuOption == 1)
        {
            cout << endl;

             //This function will get the data from the input file and puts the
             //data into the contents of the nodes, then add the nodes to the
             //empty lists passed by Reference to the function
            CreatList(head);
        }

        else if (menuOption == 2)
        {
             //This function will output the contents of the nodes of the linked
             //list created in the CreatList function
            DisplayList(head);
        }

        else if (menuOption == 3)
        {
             //This function check if the list created or modified by the user
             //is empty or no and then  outputs if the list is empty or no
            IsEmpty(head);
        }

        else if (menuOption == 4)
        {
             //This function will get a name from the user and search the name
             //content of each node to see if there is a matching name in the
             //nodes with the name searched by the user.
            SearchName(head);
        }

        else if (menuOption == 5)
        {
             //This function will ask the user which node the user wants to
             //remove and it searches the nodes based on the name content of the
             //nodes and if the name content of a node matches the name input by
             //the user, the function removes the node from the list,
            RemoveNode(head);

        }

        else if (menuOption == 6)
        {
            //This function Allows the user to delete all the nodes in the
            //linked list and make an empty list out of the list of the names
            //that we had
            ClearList(head);
        }

        // GETS USER INPUT AND CHECK IT

        menuOption = PrintMenu();
    }

}



```

```
/*************************************************************************
* AUTHOR     : Amirarsalan Valipour & Ali Eshghi
* STUDENT ID : 1103126 - 1112261
* LAB #10    : Creating an Ordered List
* CLASS      : CS 1B
* SECTION    : MW - 7:30 pm
* DUE DATE   : 11/05/2019
*************************************************************************
* PrintMenu
*************************************************************************
*  This function will output the menu, gets the user's option and checks to see
*    if the input is eligible or not.
*************************************************************************/

#include "MyHeader.h"

int PrintMenu()
{

    int  option;        //In & Calc - users choice for the menu

    bool checkInp;      //Calc      - LCV for checking the user's input

    checkInp = true;

    do
    {
        //INPUT

        cout << endl;

        cout << "LIST MENU:"             << endl;
        cout << "1 - Create List"        << endl;
        cout << "2 - Display List"       << endl;
        cout << "3 - Is the list empty?" << endl;
        cout << "4 - Search by name"     << endl;
        cout << "5 - Remove Node"        << endl;
        cout << "6 - Clear List"         << endl;
        cout << "0 - to Exit"            << endl;

        cout << "Enter a command (0 to exit): ";

        //CHECKS FOR THE CHAR INPUT

        if (!(cin >> option))
        {

            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');

            cout << endl;
            cout << "**** Please input a NUMBER between 0 and 6 ****";
            cout << endl;

            checkInp = false;

        }

        //CHECKS FOR THE RANGE ERROR

        else if (option > 6 || option < 0)
        {

            cout << endl;
            cout << "**** The number "              << option
                 << " is an invalid entry     ****" << endl;
            cout << "**** Please input a number between 0 and 6 ****";
            cout << endl;

            checkInp = false;

        }

        //PASS

        else
        {

            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            checkInp = true;

        }

    }while(!checkInp);

    return option;
}








```

```cpp
/*************************************************************************
* AUTHOR     : Amirarsalan Valipour & Ali Eshghi
* STUDENT ID : 1103126 - 1112261
* LAB #10    : Creating an Ordered List
* CLASS      : CS 1B
* SECTION    : MW - 7:30 pm
* DUE DATE   : 11/05/2019
*************************************************************************
* CreatsList
*************************************************************************
*  This function Creates a linked list adding each node in alphabetical order
*   from the input file.
*************************************************************************/

#include "MyHeader.h"

void CreatList(PersonNode *&head)
{

    fstream     inFile;

    //OPEN FILE

    inFile.open("inFile.txt");

    PersonNode *persPtr;    //In & Calc  - node to input data
    PersonNode *searchPtr;      //Calc       - node to go through the loop
    PersonNode  node;           //In         - node to set the input into main
                                //              main linked list

    bool found;                 //Proc - lcv  varibale

    persPtr   = head;

    while (inFile)
    {
        //FIRST INPUT

        getline(inFile,node.name);
        inFile.get(node.gender);
        inFile >> node.age;
        inFile.ignore(10000,'\n');

        //CREATE A NEW NODE

        persPtr   = new PersonNode;
        *persPtr  = node;
        searchPtr = head;

        found     = false;

        //IF EMPTY
        if (head == NULL)
        {

            persPtr -> next = head;
            persPtr -> prev = NULL;
            head            = persPtr;
            persPtr         = NULL;

        }

        //ADDING IN TO THE FRONT

        else if(head -> name > persPtr -> name)
        {

            persPtr -> prev = NULL;
            persPtr -> next = head;
            head -> prev    = persPtr;
            head            = persPtr;
            persPtr         = NULL;

        }

        else
        {
            //COMPARING NODES TO SEE WHICH SHOULD GO WHERE

            while ((searchPtr -> next != NULL) && !found)
            {

                if (searchPtr -> next -> name > persPtr -> name)
                {

                    found = true;

                }

                else
                {

                    searchPtr = searchPtr -> next;

                }
            }

            //TAIL

            if(searchPtr -> next == NULL)
            {

                persPtr -> next = NULL;
                persPtr -> prev = searchPtr;
                searchPtr -> next = persPtr;

                persPtr = NULL;
            }

            //MIDDLE

            else
            {

                persPtr -> next = searchPtr -> next;
                persPtr -> prev = searchPtr;
                searchPtr -> next -> prev = persPtr;
                searchPtr -> next = persPtr;

                searchPtr = persPtr = NULL;

            }


        } // END - ELSE

        cout << "Adding: " << node.name << endl;

    } // END - WHILE

    //CLOSE FILE

    inFile.close();

    cout << endl;

}
```

```cpp
/*************************************************************************
* AUTHOR     : Amirarsalan Valipour & Ali Eshghi
* STUDENT ID : 1103126 - 1112261
* LAB #10    : Creating an Ordered List
* CLASS      : CS 1B
* SECTION    : MW - 7:30 pm
* DUE DATE   : 11/05/2019
*************************************************************************
* DisplayList
*************************************************************************
*  This function will Displays the linked-list in the format described in the
*   expected input/output section on the console;.
*************************************************************************/

#include "MyHeader.h"

void DisplayList(PersonNode *head)
{
    //VARIABLES

    int i;  //Calc & Out - used for quantity

    //INITIALIZING

    i = 1;

     if (head == NULL)
     {
         cout << endl;
         cout << "Can't Display an empty list!";
         cout << endl;
     }


     else
     {
        //SETTING UP THE TABLE

        cout << endl;

        cout << right;

        cout << setw(4)  << '#';
        cout << setw(8)  << "NAME";
        cout << setw(29) << "GENDER";
        cout << setw(7)  << "AGE ";

        cout << left;

        cout << endl;

        cout << "------- ";
        cout << "------------------------  ";
        cout << "-------- ";
        cout << "-----";

        cout << endl;

        //OUTPUTTING DATA IN ALPHABETICAL ORDER

        while(head != NULL)
        {
            cout << right;
            cout << setw(4) << i << "    ";
            cout << left;
            cout << setw(29) << head -> name;
            cout << setw(7)  << head -> gender;
            cout << setw(4)  << head -> age;
            cout << endl;
            cout << left;

            i++;
            head = head -> next;
        }

     }


}


```

```
/*************************************************************************
* AUTHOR     : Amirarsalan Valipour & Ali Eshghi
* STUDENT ID : 1103126 - 1112261
* LAB #10    : Creating an Ordered List
* CLASS      : CS 1B
* SECTION    : MW - 7:30 pm
* DUE DATE   : 11/05/2019
*************************************************************************
* IsEmpty
*************************************************************************
*  This function will provides an appropriate response indicating if the list
*   is empty or not.
*************************************************************************/

#include "MyHeader.h"

void IsEmpty(PersonNode *head)
{

    //EMPTY LIST

    if (head == NULL)
    {
        cout << endl;
        cout << "Yes, the list is empty.";
        cout << endl;
    }

    //NON EMPTY LIST

    else
    {
        cout << endl;
        cout << "No, The list is NOT empty.";
        cout << endl;
    }

}
```

```
/*************************************************************************
* AUTHOR     : Amirarsalan Valipour & Ali Eshghi
* STUDENT ID : 1103126 - 1112261
* LAB #10    : Creating an Ordered List
* CLASS      : CS 1B
* SECTION    : MW - 7:30 pm
* DUE DATE   : 11/05/2019
*************************************************************************
* SearchName
*************************************************************************
*  This function Allows the user to input a name and will output the node as
*   described.
*************************************************************************/

#include "MyHeader.h"

void SearchName(PersonNode *head)
{
     PersonNode *persPtr;   //Proc - stores the target name
     PersonNode node;       //Proc & In - passes the info into persPtr

     bool found;        //Proc - condition value for the searched name

     //CHECKS FOR EMPTY LIST

     found = false;

     if (head == NULL)
     {
         cout << endl;
         cout << "Search operation is not available for an empty list.";
         cout << endl;
     }

     else
     {
         //NEW NODE

         persPtr = new PersonNode;

         //INPUT

         cout << endl;
         cout << "Who would you like to search for? ";
         getline(cin, node.name);

         *persPtr = node;

         cout << endl;
         cout << "Searching for " << persPtr -> name << "...";
         cout << endl;

         //GOES ATHROUGH THE LIST


```

```

        while ((head -> next != NULL) && !found)
        {
            //IF FOUND

            if (head -> name == persPtr -> name)
            {
                cout << endl;
                cout << "Name:    " << head -> name   << endl;
                cout << "Gender:  " << head -> gender << endl;
                cout << "Age:     " << head -> age     << endl;

                found = true;
            }

            else
            {
                head  = head -> next;
                found = false;

            }


        }

        if (head -> next == NULL && head -> name == persPtr -> name)
        {
           cout << endl;
           cout << "Name:    " << head -> name   << endl;
           cout << "Gender:  " << head -> gender << endl;
           cout << "Age:     " << head -> age     << endl;

           found = true;
        }

        //IF NOT FOUND IN THE LIST

        else if (!found)
        {
            cout << endl;
            cout << "I'm sorry, \"" << persPtr -> name;
            cout << "\" was NOT found!";
            cout << endl;
        }

    }

    persPtr = NULL;
}

```

```
/**************************************************************************
* AUTHOR     : Amirarsalan Valipour & Ali Eshghi
* STUDENT ID : 1103126 - 1112261
* LAB #10    : Creating an Ordered List
* CLASS      : CS 1B
* SECTION    : MW - 7:30 pm
* DUE DATE   : 11/05/2019
**************************************************************************
* RemoveNode
**************************************************************************
*  This function Allows the user to input a name and removes the node from the
*   list.
**************************************************************************/

#include "MyHeader.h"

void RemoveNode(PersonNode *&head)
{
     PersonNode *persPtr;   //Proc - stores the target name
     PersonNode *ptr;       //Proc - search node
     PersonNode *rmv;       //Proc - remove node
     PersonNode node;       //Proc & In - passes the info into persPtr

     bool found;        //Proc - condition value for the searched name

     //CHECKS FOR EMPTY LIST

     found = false;

     if (head == NULL)
     {
         cout << endl;
         cout << "Remove operation is not available for an empty list.";
         cout << endl;
     }

     else
     {
         //NEW NODE

         persPtr  = new PersonNode;
         ptr = head;

         //INPUT

         cout << endl;
         cout << "Who would you like to remove?  ";
         getline(cin, node.name);

         *persPtr = node;

         //FIRST CASE
         if (ptr -> next == NULL || ptr -> name == persPtr -> name)
         {
             head = ptr -> next;
```

```
            ptr -> next -> prev = NULL;
            delete ptr;


            cout << endl;
            cout << "Removing " << persPtr -> name << '!';
            cout << endl;
        }

        else
        {
            found = false;

            //GOES THROUGH
            while (ptr -> next != NULL && !found)
            {
                if (ptr -> name == persPtr -> name)
                {
                    found = true;
                }
                else
                {
                    ptr = ptr -> next;
                }
            }

            //IF FOUND DELETE

            if (found)
            {
                rmv = ptr;
                ptr = ptr -> prev;
                ptr -> next = rmv -> next;
                rmv -> next -> prev = ptr;
                delete rmv;

                rmv = NULL;


                cout << endl;
                cout << "Removing " << persPtr -> name << '!';
                cout << endl;
            }

            //DELETE OTHER
            else if (ptr -> name == persPtr -> name)
            {
                rmv = ptr;
                ptr = ptr -> prev;
                ptr -> next = NULL;
                delete rmv;

                rmv = NULL;


                cout << endl;
                cout << "Removing " << persPtr -> name << '!';
                cout << endl;
            }

            //IF THEY DON"T EXIST
            else
            {
                cout << endl;
                cout << "I'm sorry, \"" << persPtr -> name;
                cout << "\" was NOT found!";
                cout << endl;
            }

        }

        ptr = NULL;
    }
}
```

```cpp
/******************************************************************************
* AUTHOR     : Amirarsalan Valipour & Ali Eshghi
* STUDENT ID : 1103126 - 1112261
* LAB #10    : Creating an Ordered List
* CLASS      : CS 1B
* SECTION    : MW - 7:30 pm
* DUE DATE   : 11/05/2019
******************************************************************************
* ClearList
******************************************************************************
*  This function Allows the user to delete all the nodes in the linked list.
******************************************************************************/

#include "MyHeader.h"

void ClearList(PersonNode *&head)
{
    bool clear; //Calc - condition to check for empty list

    clear = false;

    //CHECKS FOR EMPTY LIST

    if (head == NULL)
    {
        cout << endl;
        cout << "The list has been cleared!";
        cout << endl;
    }

    //IF THE LIST IS NOT EMPTY

    else
    {
        cout << endl;
        cout << "CLEARING LIST:" << endl;

        while(!clear)
        {
            if(head != NULL)
            {
                cout << "Removing " << head -> name << endl;

                head = head -> next;

                delete head;

            }
            else
            {
                clear = true;
            }

        }
    }

}

```