```
/*************************************************************************
* 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;
```

```cpp
            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;
        }
}

```