```
**************************************************************
*   PROGRAMMED BY : Andrew Gharios
*   STUDENT ID    : 1449366
*   CLASS         : M-Th 5-7:20p
*   LAB #10       : Creating an ordered list
**************************************************************
STACK 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? 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

Enter a command? 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

Enter a command? 3
No, the list is NOT empty.

Enter a command? 4

Who would you like to search for? Age, Sue

```

```
Searching for Age, Sue...

Name:    Age, Sue
Gender:  F
Age:     32

Enter a command? 4

Who would you like to search for? Sassin, Anna

Searching for Sassin, Anna...

Name:    Sassin, Anna
Gender:  F
Age:     62

Enter a command? 4

Who would you like to search for? Ding, Bill

Searching for Ding, Bill...

Name:    Ding, Bill
Gender:  M
Age:     21

Enter a command? 4

Who would you like to search for? Smith, Will

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

Enter a command? 5

Who would you like to remove? Age, Sue

Searching for Age, Sue...
Removing Age, Sue!

Enter a command? 5

Who would you like to remove? Post, Mark

Searching for Post, Mark...
Removing Post, Mark!

Enter a command? 5

```

```
Who would you like to remove? Sassin, Anna

Searching for Sassin, Anna...
Removing Sassin, Anna!

Enter a command? 5

Who would you like to remove? Smith, Will

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

Enter a command? 6

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

The list has been cleared!


Enter a command? 2

Can't display an empty list

Enter a command? 3

Yes, the list is empty.

Enter a command? 4

Can't search an empty list

Enter a command? 5

Can't remove from an empty list!

Enter a command? 6

The list has been cleared!

Enter a command? x
**** Please enter a NUMBER between 0 and 6 ****

```

```
Enter a command? 7
**** The number 7 is an invalid entry     *****
**** Please input a number between 0 and 6 *****

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

Enter a command? 0
```

```
#ifndef HEADER_H_
#define HEADER_H_

#include <iostream> // cin, cout.
#include <string>   // string datatype variables.
#include <iomanip>  // fixed, setw, setprecision.
#include <fstream>
#include <limits>
#include <ios>
using namespace std;

enum Menu {
    EXIT = 0,
    CREATELIST,
    DISPLIST,
    ISEMPTY,
    SEARCH,
    REMOVE,
    CLEAR
};

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

const int INPUT_COL = 14; // CALC - setw size for display column.

/*****************************************************************************
* CreateList
*   This function will create a list and take all the data from input file.
*  ==> returns nothing.
*
  *****************************************************************************
  /
void CreateList(PersonNode*& head); // IN & CALC - List.

/*****************************************************************************
* DispList
*   This function will receive a list and display all of it.
*  ==> returns nothing.
*
  *****************************************************************************
  /
void DispList(PersonNode* head);

```

```cpp
/*******************************************************************************
* IsEmpty
*   This function will receive a list and check if it's empty.
*  ==> returns nothing.
*  ******************************************************************************/
void IsEmpty(PersonNode* head); // IN & CALC - Queue.

/*******************************************************************************
* DispList
*   This function will receive and search for a person and displays their
*   information.
*  ==> returns nothing.
*  ******************************************************************************/
void Search(PersonNode* head);

/*******************************************************************************
* Remove
*   This function will receive a list and remove a person from it.
*  ==> returns nothing.
*  ******************************************************************************/
void Remove(PersonNode*& head);

/*******************************************************************************
* Size
*   This function will receive a stack and check it's size and display it.
*  ==> returns nothing.
*  ******************************************************************************/
void Size(PersonNode* head); // IN & CALC - Queue.

/*******************************************************************************
* Clear
*   This function will receive a list of nodes and clear it.
*  ==> returns nothing.
*  ******************************************************************************/
void Clear(PersonNode*& head);

/*******************************************************************************
 * PrintHeaderFile
 *   This function will output the header information
 * ==> returns nothing.
***************************************************************************/
void PrintHeaderFile(ostream& output,      // IN  - output datatype.
    string asName,        // IN  - assignment name
    int asNum,             // IN  - assignment number
    string studentName,   // IN  - student's name
    string classInfo,     // IN  - class that is being taken
    char asType,          // IN  - assignment type
    long long studentID); // IN  - student ID

#endif


```

```cpp
#include "Header.h"

/******************************************************************************
 * PrintHeaderFile
 *   This function will output the header information
 *______________________________________________________________________________
 * PRE-CONDITIONS
 *   The following parameters need to have a defined value prior to calling
 *   the function
 *          asName: The name of the assignment given in the course
 *          asNum: The number of the assignment given in the course
 *          studentName: The name of the student writing the code
 *          classInfo: The course name, date, and time of the class
 *          asType: Will either output as a lab or an assignment
 *          studentID: The Identification Number of the student
 ******************************************************************************/

void PrintHeaderFile(ostream& output,       // IN - output datatype.
    string asName,      // IN - assignment name
    int asNum,            // IN - assignment number
    string studentName,  // IN - student's name
    string classInfo,    // IN - class that is being taken
    char asType,          // IN - assignment type
    long long studentID) // IN - student ID
{
    output << left;
    output << "***************************************************************
      \n";
    output << "*   PROGRAMMED BY : " << studentName << endl;
    output << "*    " << setw(14) << "STUDENT ID " << ": " << studentID << endl;
    output << "*    " << setw(14) << "CLASS " << ": " << classInfo << endl;
    output << "*    ";

    // PROCESSING - This will adjust setws and format appropriately based
    //               on if this is a lab 'L' or assignment

    if (toupper(asType) == 'L')
    {
        output << "LAB #" << setw(9);
    }
    else
    {
        output << "ASSIGNMENT #" << setw(2);
    }
    output << asNum << ": " << asName << endl;
    output << "***************************************************************";
    output << right << endl;

    return;
}
```

```
/
  ****************************************************************************
  ***
* AUTHOR     : Andrew Gharios
* STUDENT ID : 1449366
* LAB 10     : Creating an ordered list.
* CLASS      : CS1B
* SECTION    : M-TH: 5-7:20p
* DUE DATE   : 7/15/21
******************************************************************************
  */
#include "Header.h"

/
  ****************************************************************************
  ***
* Creating an oredered list
*-----------------------------------------------------------------------------
  -
* This program will provide a menu for the user to be able to manipulate a,
* ordered list. The user has the option to
*-----------------------------------------------------------------------------
  -
* INPUT:
* input : user menu selection.
******************************************************************************
  */
int main()
{
    /
      ************************************************************************
      ***
    * CONSTANTS
    *
      ------------------------------------------------------------------------
      -
    * OUTPUT - USED FOR CLASS HEADING
    *
      ------------------------------------------------------------------------
      -
    * PROGRAMMER : Programmer's Name
    * CLASS      : Student's Course
    * SECTION    : Class Days and Times
    * LAB_NUM    : Lab Number (specific to this lab)
    * LAB_NAME   : Title of the Lab
    **************************************************************************
      /

    const string AS_NAME = "Creating an ordered list";
```

```cpp
    const int AS_NUM = 10;
    const string STUDENT_NAME = "Andrew Gharios";
    const string CLASS_INFO = "M-Th 5-7:20p";
    const char AS_TYPE = 'L';
    const long long STUDENT_ID = 1449366;

    PersonNode* head;  // IN & CALC - Stack front.
    int  input;        // IN & CALC - menu input.
    Menu menu;         // CALC      - Menu option.
    bool invalid;      // CALC      - Validation for input.

    head = NULL;

    PrintHeaderFile(cout, AS_NAME, AS_NUM, STUDENT_NAME, CLASS_INFO, AS_TYPE, STUDENT_ID);

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

    do
    {
        do
        {
            invalid = false;
            cout << "\nEnter a command? ";
            if (!(cin >> input))
            {
                cout << "**** Please enter a NUMBER between 0 and 6 ****\n";
                cin.clear();
                cin.ignore(numeric_limits<streamsize>::max(), '\n');
                invalid = true;
            }
            else if (input < 0 || input > 6)
            {

                cout << "**** The number " << input << " is an invalid entry *****\n";
                cout << "**** Please input a number between 0 and 6 *****\n";
                invalid = true;
            }


        } while (invalid);

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

        menu = Menu(input);

        switch (menu)
        {
        case EXIT:
            break;
        case CREATELIST:
            if (head == NULL)
            {
                CreateList(head);
            }
            else
            {
                cout << "\nThere's already a created list.\n";
            }
            break;
        case DISPLIST:
            DispList(head);
            break;
        case ISEMPTY:
            IsEmpty(head);
            break;
        case SEARCH:
            Search(head);
            break;
        case REMOVE:
            Remove(head);
            break;
        case CLEAR:
            Clear(head);
            break;
        }
    } while (menu != EXIT);

    return 0;

}
```

```cpp
#include "Header.h"

/*****************************************************************************
* CreateList
*   This function will create a list and take all the data from input file.
*
* INPUTS:
*   head : list.
*
* No outputs.
*
  ****************************************************************************
  /
void CreateList(PersonNode*& head) // IN & CALC - List.
{
    PersonNode* perPtr;    // CALC - Searching pointer.
    ifstream inFile;   // CALC - Input file variable.
    PersonNode* searchPtr; // CALC - Search item.
    bool found;            // CALC - If search item was found.

    inFile.open("input.txt");

    perPtr    = NULL;
    searchPtr = NULL;
    perPtr = new PersonNode;
    found = false;


    while (inFile && perPtr != NULL)
    {
        getline(inFile, perPtr->name);
        inFile.get(perPtr->gender);
        inFile >> perPtr->age;
        inFile.ignore(numeric_limits<streamsize>::max(), '\n');

        cout << "\nAdding: " << perPtr->name;

        if (head == NULL || head->name > perPtr->name)
        {
            perPtr->next = head;
            perPtr->prev = NULL;
            if (head != NULL)
            {
                head->prev = perPtr;
            }
            head = perPtr;
        }
        else
        {
            searchPtr = head;
            while (searchPtr->next != NULL && !found)
            {
                if (searchPtr->next->name > perPtr->name)
                {
                    found = true;
                }
                else
                {
                    searchPtr = searchPtr->next;
                }
            }

            found = false;
            perPtr->next = searchPtr->next;
            perPtr->prev = searchPtr;
            if (searchPtr->next != NULL)
            {
                searchPtr->next->prev = perPtr;
            }
            searchPtr->next = perPtr;
        }

        perPtr = new PersonNode;
    }
    cout << endl;

    perPtr = NULL;
    delete perPtr;
    inFile.close();

}
```

```cpp
#include "Header.h"

/******************************************************************************
* Display List
*   This function will receive a list and display all of it.
*
* INPUTS:
*   head : list.
*
* No outputs.
*                                                                               *****************************************************************************/
void DispList(PersonNode* head) // IN & CALC - List.
{
    const int NUM_COL = 5;
    const int NAME_COL = 25;
    const int GNDR_COL = 9;

    PersonNode* perPtr; // CALC - Person pointer.
    int count;          // CALC - number of people count.

    count  = 1;
    perPtr = head;


    if (perPtr == NULL)
    {
        cout << "\nCan\'t display an empty list\n";
    }
    else
    {
        cout << left;
        cout << setw(NUM_COL) << " #";
        cout << setw(NAME_COL) << "NAME" << setw(GNDR_COL) << " GENDER" << " AGE\n";

        cout << setw(NUM_COL) << string(NUM_COL - 1, '-') << setw(NAME_COL) << string(NAME_COL - 1, '-');
        cout << setw(GNDR_COL) << string(GNDR_COL - 1, '-') << string(5, '-') << endl;
        cout << right;
    }

    while (perPtr != NULL)
    {
        cout << left;
        cout << " " << setw(NUM_COL - 1)  << count;
        cout << setw(NAME_COL) << perPtr->name;
        cout << "   " << setw(GNDR_COL - 2) << perPtr->gender;
        cout << perPtr->age << endl;
        cout << right;

        perPtr = perPtr->next;
        count++;
    }
}
```

```cpp
#include "Header.h"

/******************************************************************************
* Search
*   This function will receive a list and search for a user choosen person then
*   display that person's information.
*
* INPUTS:
*   head : list.
*
* No outputs.
* *****************************************************************************/
void Search(PersonNode* head) // IN & CALC - List.
{
    const int COL_SIZE = 9; // Setw size.

    string searchItem;     // IN/CALC - Search item.
    PersonNode* searchPtr; // CALC    - Searching pointer.
    bool found;            // CALC    - If search item was found.

    searchPtr = head;
    found     = false;

    if (head != NULL)
    {
        cout << "\nWho would you like to search for? ";
        getline(cin, searchItem);
        cout << "\nSearching for " << searchItem << "..." << endl;

        while (!found && searchPtr != NULL)
        {
            if (searchItem == searchPtr->name)
            {
                found = true;
                cout << left;
                cout << setw(COL_SIZE + 1) << "\nName: " << searchPtr->name;
                cout << endl << setw(COL_SIZE) << "Gender: " << searchPtr->gender;
                cout << endl << setw(COL_SIZE) << "Age: " << searchPtr->age << endl;
            }
            else
            {
                searchPtr = searchPtr->next;
            }
        }

        if (!found)
        {
            cout << "I\'m sorry, \"" << searchItem << "\" was NOT found!\n";
        }
        searchPtr = NULL;
    }
    else
    {
        cout << "\nCan't search an empty list\n";
    }
}
```

```cpp
#include "Header.h"

#include "Header.h"

/*************************************************************************
* IsEmpty
*   This function will receive a list and check if it's empty or not.
*
* INPUTS:
*   head : list.
*
* No outputs.
*
  ***********************************************************************
  /
void IsEmpty(PersonNode* head) // IN & CALC - List.
{
    if (head == NULL)
    {
        cout << "\nYes, the list is empty.\n";
    }
    else
    {
        cout << "\nNo, the list is NOT empty.\n";
    }
}
```

```
#include "Header.h"

/*************************************************************************
* Remove
*   This function will receive a list and remove a person based on user's input
*
* INPUTS:
*   head : list.
*
* No outputs.
*
  ***********************************************************************
  /
void Remove(PersonNode*& head) // IN & CALC - List.
{
    PersonNode* searchPtr; // CALC    - Searching pointer.
    string searchItem;     // IN/CALC - Search item.
    bool found;            // CALC    - If search item was found.

    searchPtr = NULL;
    found = false;

    if (head != NULL)
    {
        cout << "\nWho would you like to remove? ";
        getline(cin, searchItem);
        cout << "\nSearching for " << searchItem << "...\n";

        searchPtr = head;

        while (!found && searchPtr != NULL)
        {
            if (searchItem == searchPtr->name)
            {
                found = true;
                if (searchPtr == head)
                {
                    head = searchPtr->next;
                    head->prev = NULL;
                }
                else if (searchPtr->next == NULL)
                {
                    searchPtr->prev->next = NULL;
                }
                else
                {
                    searchPtr->next->prev = searchPtr->prev;
                    searchPtr->prev->next = searchPtr->next;
                }
```

```

                    searchPtr = NULL;
                    delete searchPtr;

                    cout << "Removing " << searchItem << "!\n";
                }
                else
                {
                    searchPtr = searchPtr->next;
                }
            }

            if (!found)
            {
                cout << "I'm sorry, \"" << searchItem << "\" was NOT found!\n";
            }
        }
        else
        {
            cout << "\nCan't remove from an empty list!\n";
        }

}
```

```cpp
#include "Header.h"

/******************************************************************************
* Clear
*   This function will receive a list of nodes and clear it.
*
*  INPUTS:
*  head : list.
*
*  No outputs.
******************************************************************************/
void Clear(PersonNode*& head) // IN & CALC - List.
{
    PersonNode* searchPtr; // CALC    - Searching pointer.

    searchPtr = head;

    if (searchPtr == NULL)
    {
        cout << "\nThe list has been cleared!\n";
    }
    else
    {
        cout << "\nCLEARING LIST:\n";
        while (searchPtr != NULL)
        {
            cout << "Removing " << searchPtr->name << "!\n";
            head = searchPtr->next;
            delete searchPtr;
            searchPtr = head;
        }
        searchPtr = NULL;
        cout << endl;
        cout << "The list has been cleared!\n" << endl;
    }
}
```