

iostream.txt

```
1 *****
2 * PROGRAMMED BY : Ali Eshghi & Amirarsalan Valipour
3 * CLASS          : CS1B
4 * SECTION        : MW: 7:30p - 9:50p
5 * LAB #10        : Creating an Ordered List
6 *****
7
8 LIST MENU:
9 1 - Create List
10 2 - Display List
11 3 - Is the list empty?
12 4 - Search by name
13 5 - Remove Node
14 6 - Clear List
15 0 - to Exit
16 Enter a command (0 to exit): 1
17
18 Adding: Payne, Royal
19 Adding: Ding, Bill
20 Adding: Post, Mark
21 Adding: Sassin, Anna
22 Adding: Lear, Shanda
23 Adding: Longbottom, Iva
24 Adding: Dwyer, Barb
25 Adding: Hogg, Ima
26 Adding: Belcher, Ura
27 Adding: Age, Sue
28
29
30 LIST MENU:
31 1 - Create List
32 2 - Display List
33 3 - Is the list empty?
34 4 - Search by name
35 5 - Remove Node
36 6 - Clear List
37 0 - to Exit
38 Enter a command (0 to exit): 2
39
40  #    NAME                                GENDER    AGE
41  ----
42  1    Age, Sue                            F          32
43  2    Belcher, Ura                        F          46
44  3    Ding, Bill                          M          21
45  4    Dwyer, Barb                         F          24
46  5    Hogg, Ima                           F          43
47  6    Lear, Shanda                        F          18
48  7    Longbottom, Iva                    F          45
49  8    Payne, Royal                        M          73
50  9    Post, Mark                          M          20
51 10    Sassin, Anna                        F          62
52
53 LIST MENU:
54 1 - Create List
55 2 - Display List
```

```
56 3 - Is the list empty?
57 4 - Search by name
58 5 - Remove Node
59 6 - Clear List
60 0 - to Exit
61 Enter a command (0 to exit): 3
62
63 No, The list is NOT empty.
64
65 LIST MENU:
66 1 - Create List
67 2 - Display List
68 3 - Is the list empty?
69 4 - Search by name
70 5 - Remove Node
71 6 - Clear List
72 0 - to Exit
73 Enter a command (0 to exit): 4
74
75 Who would you like to search for? Age, Sue
76
77 Searching for Age, Sue...
78
79 Name:    Age, Sue
80 Gender:  F
81 Age:     32
82
83 LIST MENU:
84 1 - Create List
85 2 - Display List
86 3 - Is the list empty?
87 4 - Search by name
88 5 - Remove Node
89 6 - Clear List
90 0 - to Exit
91 Enter a command (0 to exit): 4
92
93 Who would you like to search for? Sassin, Anna
94
95 Searching for Sassin, Anna...
96
97 Name:    Sassin, Anna
98 Gender:  F
99 Age:     62
100
101 LIST MENU:
102 1 - Create List
103 2 - Display List
104 3 - Is the list empty?
105 4 - Search by name
106 5 - Remove Node
107 6 - Clear List
108 0 - to Exit
109 Enter a command (0 to exit): 4
110
```

```
111 Who would you like to search for? Ding, Bill
112
113 Searching for Ding, Bill...
114
115 Name:    Ding, Bill
116 Gender:  M
117 Age:     21
118
119 LIST MENU:
120 1 - Create List
121 2 - Display List
122 3 - Is the list empty?
123 4 - Search by name
124 5 - Remove Node
125 6 - Clear List
126 0 - to Exit
127 Enter a command (0 to exit): 4
128
129 Who would you like to search for? Smith, Will
130
131 Searching for Smith, Will...
132
133 I'm sorry, "Smith, Will" was NOT found!
134
135 LIST MENU:
136 1 - Create List
137 2 - Display List
138 3 - Is the list empty?
139 4 - Search by name
140 5 - Remove Node
141 6 - Clear List
142 0 - to Exit
143 Enter a command (0 to exit): 5
144
145 Who would you like to remove? Age, Sue
146
147 Removing Age, Sue!
148
149 LIST MENU:
150 1 - Create List
151 2 - Display List
152 3 - Is the list empty?
153 4 - Search by name
154 5 - Remove Node
155 6 - Clear List
156 0 - to Exit
157 Enter a command (0 to exit): 5
158
159 Who would you like to remove? Post, Mark
160
161 Removing Post, Mark!
162
163 LIST MENU:
164 1 - Create List
165 2 - Display List
```

```
166 3 - Is the list empty?
167 4 - Search by name
168 5 - Remove Node
169 6 - Clear List
170 0 - to Exit
171 Enter a command (0 to exit): 5
172
173 Who would you like to remove? Sassin, Anna
174
175 Removing Sassin, Anna!
176
177 LIST MENU:
178 1 - Create List
179 2 - Display List
180 3 - Is the list empty?
181 4 - Search by name
182 5 - Remove Node
183 6 - Clear List
184 0 - to Exit
185 Enter a command (0 to exit): 5
186
187 Who would you like to remove? Smith, Will
188
189 I'm sorry, "Smith, Will" was NOT found!
190
191 LIST MENU:
192 1 - Create List
193 2 - Display List
194 3 - Is the list empty?
195 4 - Search by name
196 5 - Remove Node
197 6 - Clear List
198 0 - to Exit
199 Enter a command (0 to exit): 6
200
201 CLEARING LIST:
202 Removing Belcher, Ura
203 Removing Ding, Bill
204 Removing Dwyer, Barb
205 Removing Hogg, Ima
206 Removing Lear, Shanda
207 Removing Longbottom, Iva
208 Removing Payne, Royal
209
210 LIST MENU:
211 1 - Create List
212 2 - Display List
213 3 - Is the list empty?
214 4 - Search by name
215 5 - Remove Node
216 6 - Clear List
217 0 - to Exit
218 Enter a command (0 to exit): 2
219
220 Can't Display an empty list!
```

```
221
222 LIST MENU:
223 1 - Create List
224 2 - Display List
225 3 - Is the list empty?
226 4 - Search by name
227 5 - Remove Node
228 6 - Clear List
229 0 - to Exit
230 Enter a command (0 to exit): 3
231
232 Yes, the list is empty.
233
234 LIST MENU:
235 1 - Create List
236 2 - Display List
237 3 - Is the list empty?
238 4 - Search by name
239 5 - Remove Node
240 6 - Clear List
241 0 - to Exit
242 Enter a command (0 to exit): 4
243
244 Search operation is not available for an empty list.
245
246 LIST MENU:
247 1 - Create List
248 2 - Display List
249 3 - Is the list empty?
250 4 - Search by name
251 5 - Remove Node
252 6 - Clear List
253 0 - to Exit
254 Enter a command (0 to exit): 5
255
256 Remove operation is not available for an empty list.
257
258 LIST MENU:
259 1 - Create List
260 2 - Display List
261 3 - Is the list empty?
262 4 - Search by name
263 5 - Remove Node
264 6 - Clear List
265 0 - to Exit
266 Enter a command (0 to exit): 6
267
268 The list has been cleared!
269
270 LIST MENU:
271 1 - Create List
272 2 - Display List
273 3 - Is the list empty?
274 4 - Search by name
275 5 - Remove Node
```

```
276 6 - Clear List
277 0 - to Exit
278 Enter a command (0 to exit): x
279
280 **** Please input a NUMBER between 0 and 6 ****
281
282 LIST MENU:
283 1 - Create List
284 2 - Display List
285 3 - Is the list empty?
286 4 - Search by name
287 5 - Remove Node
288 6 - Clear List
289 0 - to Exit
290 Enter a command (0 to exit): 7
291
292 **** The number 7 is an invalid entry      ****
293 **** Please input a number between 0 and 6 ****
294
295 LIST MENU:
296 1 - Create List
297 2 - Display List
298 3 - Is the list empty?
299 4 - Search by name
300 5 - Remove Node
301 6 - Clear List
302 0 - to Exit
303 Enter a command (0 to exit): -1
304
305 **** The number -1 is an invalid entry      ****
306 **** Please input a number between 0 and 6 ****
307
308 LIST MENU:
309 1 - Create List
310 2 - Display List
311 3 - Is the list empty?
312 4 - Search by name
313 5 - Remove Node
314 6 - Clear List
315 0 - to Exit
316 Enter a command (0 to exit): 0
317
```

MyHeader.h

```
1 /*****
2 * AUTHOR      : Amirarsalan Valipour & Ali Eshghi
3 * STUDENT ID  : 1103126 - 1112261
4 * LAB #10    : Creating an Ordered List
5 * CLASS      : CS 1B
6 * SECTION    : MW - 7:30 pm
7 * DUE DATE   : 11/05/2019
8 *****/
9 * PrintMenu
10 *****/
11 * This function will output the menu, gets the user's option and checks to see
12 *   if the input is eligible or not.
13 *****/
14
15 #ifndef MYHEADER_H_
16 #define MYHEADER_H_
17
18 #include<iostream>
19 #include<iomanip>
20 #include<string>
21 #include<fstream>
22 #include<limits>
23 #include<sstream>
24 using namespace std;
25
26 const string NAME = "Creating an Ordered List";
27 const char   TYPE  = 'L';
28 const int    NUM   = 10;
29 const string CLASS = "CS1B";
30 const string SECTION = "MW: 7:30p - 9:50p";
31
32
33
34 enum MenuOption
35 {
36     EXIT,
37     CREATE,
38     DISPLAY,
39     ISEMPY,
40     SEARCH,
41     REMOVE,
42     CLEAR
43 };
44
45 struct PersonNode
46 {
47     string    name;
48     char      gender;
49     int       age;
50     PersonNode *next;
51     PersonNode *prev;
52 };
53
54 /*****
55 * Function - PrintHeaderFile
```

MyHeader.h

```
56 * -----
57 * This function will output the class heading to the screen.
58 *
59 * return type - nothing
60 *           the function is void type
61 *****/
62 void PrintHeaderFile();
63
64 /*****
65 * Function - PrintMenu
66 * -----
67 * This function will output menu option to the screen and waits for the user to
68 * input an option to what to what to do. The options are as following:
69 *
70 * 1 - Create List
71 * 2 - Display List
72 * 3 - Is the list empty?
73 * 4 - Search by name
74 * 5 - Remove Node
75 * 6 - Clear List
76 * 0 - to Exit
77 *
78 * return type - Integer
79 *****/
80 int PrintMenu();
81
82 /*****
83 * Function - CreatList
84 * -----
85 * This function will get the data from the input file and puts the data into
86 * the contents of the nodes, then add the nodes to the empty lists passed by
87 * Reference to the function
88 *
89 * return type - nothing
90 *           the function is void type
91 *****/
92 void CreatList(PersonNode *&head);
93
94 /*****
95 * Function - DisplayList
96 * -----
97 * This function will output the contents of the nodes of the linked list
98 * created in the CreatList function
99 *
100 * return type - nothing
101 *           the function is void type
102 *****/
103 void DisplayList(PersonNode *head);
104
105 /*****
106 * Function - IsEmpty
107 * -----
108 * This function check if the list created or modified by the user is empty or
109 * no and then outputs if the list is empty or no
110 *
```


MyHeader.h

```
111 * return type - nothing
112 *           the function is void type
113 *****/
114 void IsEmpty(PersonNode *head);
115
116 /*****
117 * Function - SearchName
118 * -----
119 * This function will get a name from the user and search the name content of
120 * each node to see if there is a matching name in the nodes with the name
121 * searched by the user. if found, the function prints out every content of the
122 * node for the user, if not found, the function outputs that the name searched
123 * by the user was not found.
124 *
125 * return type - nothing
126 *           the function is void type
127 *****/
128 void SearchName(PersonNode *head);
129
130 /*****
131 * Function - RemoveNode
132 * -----
133 * This function will ask the user which node the user wants to remove and it
134 * searches the nodes based on the name content of the nodes and if the name
135 * content of a node matches the name input by the user, the function removes
136 * the node from the list, if not, the function outputs that the name searched
137 * by the user has not found in the list.
138 *
139 * return type - nothing
140 *           the function is void type
141 *****/
142 void RemoveNode(PersonNode *&head);
143
144 /*****
145 * Function - ClearList
146 *****/
147 * This function Allows the user to delete all the nodes in the linked list and
148 * make an empty list out of the list of the names that we had
149 *
150 * return type - nothing
151 *           the function is void type
152 *****/
153 void ClearList(PersonNode *&head);
154
155
156 #endif /* MYHEADER_H_ */
157
```

PrintHeaderFile.cpp

```
1 /*****
2 * AUTHOR      : Amirarsalan Valipour & Ali Eshghi
3 * STUDENT ID  : 1103126 - 1112261
4 * LAB #10     : Creating an Ordered List
5 * CLASS       : CS 1B
6 * SECTION     : MW - 7:30 pm
7 * DUE DATE    : 11/05/2019
8 *****/
9
10 #include "MyHeader.h"
11
12
13 /*****
14 * Function : PrintHeaderFile
15 * -----
16 * This function will output the class heading to the screen.
17 *****/
18
19 void PrintHeaderFile()
20 {
21
22     /*****
23     * OUTPUT - outputs class heading to file and then console.
24     *****/
25
26     cout << left;
27     cout << "*****\n" ;
28     cout << "* PROGRAMMED BY : Ali Eshghi & Amirarsalan Valipour " ;
29     cout << "\n* " << setw(14) << "CLASS" << ": " << CLASS ;
30     cout << "\n* " << setw(14) << "SECTION" << ": " << SECTION ;
31     cout << "\n* LAB #" << setw(9) << NUM << ": " << NAME ;
32     cout << "\n*****\n\n" ;
33     cout << right;
34
35 }
36
37
```

main.cpp

```

1 /*****
2 * AUTHOR      : Amirarsalan Valipour & Ali Eshghi
3 * STUDENT ID  : 1103126 - 1112261
4 * LAB #10     : Creating an Ordered List
5 * CLASS       : CS 1B
6 * SECTION     : MW - 7:30 pm
7 * DUE DATE    : 11/05/2019
8 *****/
9
10
11 #include "MyHeader.h"
12
13 /*****
14 * LAB 10
15 * -----
16 * This program gets an integer as an option from the user based in the printed
17 * menu. the first option gets the data from an input file and put the data into
18 * the content of nodes and add them to an empty list that has been created in
19 * main. the second option displays the list that has been created in the
20 * first option of the menu on the screen. the second option runs a function that
21 * determines that the list created or modified by the user is empty or no, and
22 * it outputs on the screen if the list is empty or no. the fourth option let the
23 * user to search the nodes based on the name content of the nodes and if the
24 * name content of a node matches the name content of the searched name by the
25 * user, it shows every content of the node. the fifth option will ask the user
26 * which node the user wants to remove and it searches the nodes based on the
27 * name content of the nodes and if the name content of a node matches the name
28 * input by the user, the function removes the node from the list, if not, the
29 * function outputs that the name searched by the user has not found in the list.
30 * the sixth option This function Allows the user to delete all the nodes in the
31 * linked list and make an empty list out of the list of the names that we had.
32 * -----
33 * INPUT  : option      -> menu option
34 * -----
35 * PROCESS: creating the list
36 *          Displaying the list
37 *          Check if the list is empty
38 *          Searching by the name
39 *          searching and removing a name
40 *          clearing the list
41 *
42 *
43 * -----
44 * OUTPUT : Content of the nodes
45 *          Information about the list based on the creating and modifying
46 *****/
47
48
49 int main()
50 {
51     /*****
52      * VARIABLES *
53      *****/
54
55     int menuOption; // IN - user input

```

main.cpp

```
56  PersonNode *head; // PROCESS – pointer for an empty list
57
58  //creating an empty list
59  head = NULL;
60
61
62
63  //This function outputs the class heading
64  PrintHeaderFile();
65
66
67  // GETS USER INPUT AND CHECK IT
68
69  //This function will output menu option to the screen and waits for the user
70  //to input an option to what to what to do.
71  menuOption = PrintMenu();
72
73  while (menuOption != 0)
74  {
75      if (menuOption == 1)
76      {
77          cout << endl;
78
79          //This function will get the data from the input file and puts the
80          //data into the contents of the nodes, then add the nodes to the
81          //empty lists passed by Reference to the function
82          CreatList(head);
83      }
84
85      else if (menuOption == 2)
86      {
87          //This function will output the contents of the nodes of the linked
88          //list created in the CreatList function
89          DisplayList(head);
90      }
91
92      else if (menuOption == 3)
93      {
94          //This function check if the list created or modified by the user
95          //is empty or no and then outputs if the list is empty or no
96          IsEmpty(head);
97      }
98
99      else if (menuOption == 4)
100     {
101         //This function will get a name from the user and search the name
102         //content of each node to see if there is a matching name in the
103         //nodes with the name searched by the user.
104         SearchName(head);
105     }
106
107     else if (menuOption == 5)
108     {
109         //This function will ask the user which node the user wants to
110         //remove and it searches the nodes based on the name content of the
```

main.cpp

```
111         //nodes and if the name content of a node matches the name input by
112         //the user, the function removes the node from the list,
113         RemoveNode(head);
114
115     }
116
117     else if (menuOption == 6)
118     {
119         //This function Allows the user to delete all the nodes in the
120         //linked list and make an empty list out of the list of the names
121         //that we had
122         ClearList(head);
123     }
124
125     // GETS USER INPUT AND CHECK IT
126
127     menuOption = PrintMenu();
128 }
129
130 }
131
132
133
```

PrintMenu.cpp

```
1 /*****
2 * AUTHOR      : Amirarsalan Valipour & Ali Eshghi
3 * STUDENT ID  : 1103126 - 1112261
4 * LAB #10    : Creating an Ordered List
5 * CLASS      : CS 1B
6 * SECTION    : MW - 7:30 pm
7 * DUE DATE   : 11/05/2019
8 *****/
9 * PrintMenu
10 *****/
11 * This function will output the menu, gets the user's option and checks to see
12 *   if the input is eligible or not.
13 *****/
14
15 #include "MyHeader.h"
16
17 int PrintMenu()
18 {
19
20     int option;           //In & Calc - users choice for the menu
21
22     bool checkInp;        //Calc - LCV for checking the user's input
23
24     checkInp = true;
25
26     do
27     {
28         //INPUT
29
30         cout << endl;
31
32         cout << "LIST MENU:" << endl;
33         cout << "1 - Create List" << endl;
34         cout << "2 - Display List" << endl;
35         cout << "3 - Is the list empty?" << endl;
36         cout << "4 - Search by name" << endl;
37         cout << "5 - Remove Node" << endl;
38         cout << "6 - Clear List" << endl;
39         cout << "0 - to Exit" << endl;
40
41         cout << "Enter a command (0 to exit): ";
42
43         //CHECKS FOR THE CHAR INPUT
44
45         if (!(cin >> option))
46         {
47
48             cin.clear();
49             cin.ignore(numeric_limits<streamsize>::max(), '\n');
50
51             cout << endl;
52             cout << "**** Please input a NUMBER between 0 and 6 ****";
53             cout << endl;
54
55             checkInp = false;
```

PrintMenu.cpp

```
56
57     }
58
59     //CHECKS FOR THE RANGE ERROR
60
61     else if (option > 6 || option < 0)
62     {
63
64         cout << endl;
65         cout << "**** The number " << option
66              << " is an invalid entry ****" << endl;
67         cout << "**** Please input a number between 0 and 6 ****";
68         cout << endl;
69
70         checkInp = false;
71
72     }
73
74     //PASS
75
76     else
77     {
78
79         cin.ignore(numeric_limits<streamsize>::max(), '\n');
80         checkInp = true;
81
82     }
83
84     }while(!checkInp);
85
86     return option;
87 }
88
89
90
91
92
93
94
95
```

CreatList.cpp

```

1 /*****
2 * AUTHOR      : Amirarsalan Valipour & Ali Eshghi
3 * STUDENT ID  : 1103126 - 1112261
4 * LAB #10    : Creating an Ordered List
5 * CLASS      : CS 1B
6 * SECTION    : MW - 7:30 pm
7 * DUE DATE   : 11/05/2019
8 *****/
9 * CreatsList
10 *****/
11 * This function Creates a linked list adding each node in alphabetical order
12 * from the input file.
13 *****/
14
15 #include "MyHeader.h"
16
17 void CreatList(PersonNode *&head)
18 {
19
20     fstream    inFile;
21
22     //OPEN FILE
23
24     inFile.open("inFile.txt");
25
26     PersonNode *persPtr;    //In & Calc - node to input data
27     PersonNode *searchPtr;  //Calc - node to go through the loop
28     PersonNode node;        //In - node to set the input into main
29                             // main linked list
30
31     bool found;             //Proc - lcv varibale
32
33     persPtr = head;
34
35     while (inFile)
36     {
37         //FIRST INPUT
38
39         getline(inFile, node.name);
40         inFile.get(node.gender);
41         inFile >> node.age;
42         inFile.ignore(10000, '\n');
43
44         //CREATE A NEW NODE
45
46         persPtr = new PersonNode;
47         *persPtr = node;
48         searchPtr = head;
49
50         found = false;
51
52         //IF EMPTY
53         if (head == NULL)
54         {
55

```


CreatList.cpp

```
56     persPtr -> next = head;
57     persPtr -> prev = NULL;
58     head          = persPtr;
59     persPtr        = NULL;
60
61 }
62
63 //ADDING IN TO THE FRONT
64
65 else if(head -> name > persPtr -> name)
66 {
67
68     persPtr -> prev = NULL;
69     persPtr -> next = head;
70     head -> prev    = persPtr;
71     head           = persPtr;
72     persPtr        = NULL;
73
74 }
75
76 else
77 {
78     //COMPARING NODES TO SEE WHICH SHOULD GO WHERE
79
80     while ((searchPtr -> next != NULL) && !found)
81     {
82
83         if (searchPtr -> next -> name > persPtr -> name)
84         {
85
86             found = true;
87
88         }
89
90         else
91         {
92
93             searchPtr = searchPtr -> next;
94
95         }
96     }
97
98     //TAIL
99
100     if(searchPtr -> next == NULL)
101     {
102
103         persPtr -> next = NULL;
104         persPtr -> prev = searchPtr;
105         searchPtr -> next = persPtr;
106
107         persPtr = NULL;
108     }
109
110     //MIDDLE
```

CreatList.cpp

```
111
112     else
113     {
114
115         persPtr -> next = searchPtr -> next;
116         persPtr -> prev = searchPtr;
117         searchPtr -> next -> prev = persPtr;
118         searchPtr -> next = persPtr;
119
120         searchPtr = persPtr = NULL;
121
122     }
123
124 } // END - ELSE
125
126 cout << "Adding: " << node.name << endl;
127
128 } // END - WHILE
129
130 //CLOSE FILE
131
132 inFile.close();
133
134 cout << endl;
135
136 }
137 }
138
139
140
141
142
```

DisplayList.cpp

```

1 /*****
2 * AUTHOR      : Amirarsalan Valipour & Ali Eshghi
3 * STUDENT ID  : 1103126 - 1112261
4 * LAB #10    : Creating an Ordered List
5 * CLASS      : CS 1B
6 * SECTION    : MW - 7:30 pm
7 * DUE DATE   : 11/05/2019
8 *****/
9 * DisplayList
10 *****/
11 * This function will Displays the linked-list in the format described in the
12 * expected input/output section on the console;.
13 *****/
14
15 #include "MyHeader.h"
16
17 void DisplayList(PersonNode *head)
18 {
19     //VARIABLES
20
21     int i; //Calc & Out - used for quantity
22
23     //INITIALIZING
24
25     i = 1;
26
27     if (head == NULL)
28     {
29         cout << endl;
30         cout << "Can't Display an empty list!";
31         cout << endl;
32     }
33
34
35     else
36     {
37         //SETTING UP THE TABLE
38
39         cout << endl;
40
41         cout << right;
42
43         cout << setw(4) << '#';
44         cout << setw(8) << "NAME";
45         cout << setw(29) << "GENDER";
46         cout << setw(7) << "AGE ";
47
48         cout << left;
49
50         cout << endl;
51
52         cout << "----- ";
53         cout << "----- ";
54         cout << "----- ";
55         cout << "-----";

```

DisplayList.cpp

```
56
57     cout << endl;
58
59     //OUTPUTTING DATA IN ALPHABETICAL ORDER
60
61     while(head != NULL)
62     {
63         cout << right;
64         cout << setw(4) << i << "    ";
65         cout << left;
66         cout << setw(29) << head -> name;
67         cout << setw(7) << head -> gender;
68         cout << setw(4) << head -> age;
69         cout << endl;
70         cout << left;
71
72         i++;
73         head = head -> next;
74     }
75
76 }
77
78
79 }
80
81
```

IsEmpty.cpp

```
1 /*****
2 * AUTHOR      : Amirarsalan Valipour & Ali Eshghi
3 * STUDENT ID  : 1103126 - 1112261
4 * LAB #10    : Creating an Ordered List
5 * CLASS      : CS 1B
6 * SECTION    : MW - 7:30 pm
7 * DUE DATE   : 11/05/2019
8 *****/
9 * IsEmpty
10 *****/
11 * This function will provides an appropriate response indicating if the list
12 * is empty or not.
13 *****/
14
15 #include "MyHeader.h"
16
17 void IsEmpty(PersonNode *head)
18 {
19
20     //EMPTY LIST
21
22     if (head == NULL)
23     {
24         cout << endl;
25         cout << "Yes, the list is empty.";
26         cout << endl;
27     }
28
29     //NON EMPTY LIST
30
31     else
32     {
33         cout << endl;
34         cout << "No, The list is NOT empty.";
35         cout << endl;
36     }
37 }
38 }
39
```

SearchName.cpp

```
1 /*****
2 * AUTHOR      : Amirarsalan Valipour & Ali Eshghi
3 * STUDENT ID  : 1103126 - 1112261
4 * LAB #10     : Creating an Ordered List
5 * CLASS      : CS 1B
6 * SECTION    : MW - 7:30 pm
7 * DUE DATE   : 11/05/2019
8 *****/
9 * SearchName
10 *****/
11 * This function Allows the user to input a name and will output the node as
12 * described.
13 *****/
14
15 #include "MyHeader.h"
16
17 void SearchName(PersonNode *head)
18 {
19     PersonNode *persPtr;    //Proc - stores the target name
20     PersonNode node;        //Proc & In - passes the info into persPtr
21
22     bool found;             //Proc - condition value for the searched name
23
24     //CHECKS FOR EMPTY LIST
25
26     found = false;
27
28     if (head == NULL)
29     {
30         cout << endl;
31         cout << "Search operation is not available for an empty list.";
32         cout << endl;
33     }
34
35     else
36     {
37         //NEW NODE
38
39         persPtr = new PersonNode;
40
41         //INPUT
42
43         cout << endl;
44         cout << "Who would you like to search for? ";
45         getline(cin, node.name);
46
47         *persPtr = node;
48
49         cout << endl;
50         cout << "Searching for " << persPtr -> name << "...";
51         cout << endl;
52
53         //GOES ATHROUGH THE LIST
54
55     }
```

SearchName.cpp

```

56
57 while ((head -> next != NULL) && !found)
58 {
59     //IF FOUND
60
61     if (head -> name == persPtr -> name)
62     {
63         cout << endl;
64         cout << "Name:    " << head -> name << endl;
65         cout << "Gender:  " << head -> gender << endl;
66         cout << "Age:     " << head -> age << endl;
67
68         found = true;
69     }
70
71     else
72     {
73         head = head -> next;
74         found = false;
75     }
76
77
78
79 }
80
81 if (head -> next == NULL && head -> name == persPtr -> name)
82 {
83     cout << endl;
84     cout << "Name:    " << head -> name << endl;
85     cout << "Gender:  " << head -> gender << endl;
86     cout << "Age:     " << head -> age << endl;
87
88     found = true;
89 }
90
91 //IF NOT FOUND IN THE LIST
92
93 else if (!found)
94 {
95     cout << endl;
96     cout << "I'm sorry, \"" << persPtr -> name;
97     cout << "\" was NOT found!";
98     cout << endl;
99 }
100
101 }
102
103 persPtr = NULL;
104 }
105

```

RemoveNode.cpp

```
1 /*****
2 * AUTHOR      : Amirarsalan Valipour & Ali Eshghi
3 * STUDENT ID  : 1103126 - 1112261
4 * LAB #10    : Creating an Ordered List
5 * CLASS      : CS 1B
6 * SECTION    : MW - 7:30 pm
7 * DUE DATE   : 11/05/2019
8 *****/
9 * RemoveNode
10 *****/
11 * This function Allows the user to input a name and removes the node from the
12 * list.
13 *****/
14
15 #include "MyHeader.h"
16
17 void RemoveNode(PersonNode *&head)
18 {
19     PersonNode *persPtr;    //Proc - stores the target name
20     PersonNode *ptr;        //Proc - search node
21     PersonNode *rmv;        //Proc - remove node
22     PersonNode node;        //Proc & In - passes the info into persPtr
23
24     bool found;             //Proc - condition value for the searched name
25
26     //CHECKS FOR EMPTY LIST
27
28     found = false;
29
30     if (head == NULL)
31     {
32         cout << endl;
33         cout << "Remove operation is not available for an empty list.";
34         cout << endl;
35     }
36
37     else
38     {
39         //NEW NODE
40
41         persPtr = new PersonNode;
42         ptr = head;
43
44         //INPUT
45
46         cout << endl;
47         cout << "Who would you like to remove? ";
48         getline(cin, node.name);
49
50         *persPtr = node;
51
52         //FIRST CASE
53         if (ptr -> next == NULL || ptr -> name == persPtr -> name)
54         {
55             head = ptr -> next;
```


RemoveNode.cpp

```

56     ptr -> next -> prev = NULL;
57     delete ptr;
58
59
60     cout << endl;
61     cout << "Removing " << persPtr -> name << '!';
62     cout << endl;
63 }
64
65 else
66 {
67     found = false;
68
69     //GOES THROUGH
70     while (ptr -> next != NULL && !found)
71     {
72         if (ptr -> name == persPtr -> name)
73         {
74             found = true;
75         }
76         else
77         {
78             ptr = ptr -> next;
79         }
80     }
81
82     //IF FOUND DELETE
83
84     if (found)
85     {
86         rmv = ptr;
87         ptr = ptr -> prev;
88         ptr -> next = rmv -> next;
89         rmv -> next -> prev = ptr;
90         delete rmv;
91
92         rmv = NULL;
93
94
95         cout << endl;
96         cout << "Removing " << persPtr -> name << '!';
97         cout << endl;
98     }
99
100    //DELETE OTHER
101    else if (ptr -> name == persPtr -> name)
102    {
103        rmv = ptr;
104        ptr = ptr -> prev;
105        ptr -> next = NULL;
106        delete rmv;
107
108        rmv = NULL;
109
110

```

RemoveNode.cpp

```
111         cout << endl;
112         cout << "Removing " << persPtr -> name << '!';
113         cout << endl;
114     }
115
116     //IF THEY DON'T EXIST
117     else
118     {
119         cout << endl;
120         cout << "I'm sorry, \"" << persPtr -> name;
121         cout << "\" was NOT found!";
122         cout << endl;
123     }
124 }
125
126 ptr = NULL;
127 }
128 }
129 }
130 }
```

ClearList.cpp

```
1 /*****
2 * AUTHOR      : Amirarsalan Valipour & Ali Eshghi
3 * STUDENT ID  : 1103126 - 1112261
4 * LAB #10     : Creating an Ordered List
5 * CLASS      : CS 1B
6 * SECTION     : MW - 7:30 pm
7 * DUE DATE    : 11/05/2019
8 *****/
9 * ClearList
10 *****/
11 * This function Allows the user to delete all the nodes in the linked list.
12 *****/
13
14 #include "MyHeader.h"
15
16 void ClearList(PersonNode *&head)
17 {
18     bool clear; //Calc - condition to check for empty list
19
20     clear = false;
21
22     //CHECKS FOR EMPTY LIST
23
24     if (head == NULL)
25     {
26         cout << endl;
27         cout << "The list has been cleared!";
28         cout << endl;
29     }
30
31     //IF THE LIST IS NOT EMPTY
32
33     else
34     {
35         cout << endl;
36         cout << "CLEARING LIST:" << endl;
37
38         while(!clear)
39         {
40             if(head != NULL)
41             {
42                 cout << "Removing " << head -> name << endl;
43
44                 head = head -> next;
45
46                 delete head;
47
48             }
49             else
50             {
51                 clear = true;
52             }
53         }
54     }
55 }
```

ClearList.cpp

```
56  
57 }  
58
```