

```
1 *****
2 *   PROGRAMMED BY : Andrew Gharios
3 *   STUDENT ID    : 1449366
4 *   CLASS         : M-Th 5-7:20p
5 *   LAB #10       : Creating an ordered list
6 *****
7 STACK MENU:
8 1 - Create List
9 2 - Display List
10 3 - Is the list empty?
11 4 - Search by name
12 5 - Remove Node
13 6 - Clear List
14 0 - to Exit
15
16 Enter a command? 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 Enter a command? 2
30 #   NAME                                GENDER   AGE
31 ----
32 1   Age, Sue                            F         32
33 2   Belcher, Ura                        F         46
34 3   Ding, Bill                          M         21
35 4   Dwyer, Barb                         F         24
36 5   Hogg, Ima                           F         43
37 6   Lear, Shanda                        F         18
38 7   Longbottom, Iva                     F         45
39 8   Payne, Royal                        M         73
40 9   Post, Mark                          M         20
41 10  Sassin, Anna                        F         62
42
43 Enter a command? 3
44 No, the list is NOT empty.
45
46 Enter a command? 4
47
48 Who would you like to search for? Age, Sue
49
```

```
50 Searching for Age, Sue...
51
52 Name:    Age, Sue
53 Gender:  F
54 Age:     32
55
56 Enter a command? 4
57
58 Who would you like to search for? Sassin, Anna
59
60 Searching for Sassin, Anna...
61
62 Name:    Sassin, Anna
63 Gender:  F
64 Age:     62
65
66 Enter a command? 4
67
68 Who would you like to search for? Ding, Bill
69
70 Searching for Ding, Bill...
71
72 Name:    Ding, Bill
73 Gender:  M
74 Age:     21
75
76 Enter a command? 4
77
78 Who would you like to search for? Smith, Will
79
80 Searching for Smith, Will...
81 I'm sorry, "Smith, Will" was NOT found !
82
83 Enter a command? 5
84
85 Who would you like to remove? Age, Sue
86
87 Searching for Age, Sue...
88 Removing Age, Sue!
89
90 Enter a command? 5
91
92 Who would you like to remove? Post, Mark
93
94 Searching for Post, Mark...
95 Removing Post, Mark!
96
97 Enter a command? 5
98
```

```
100 99 Who would you like to remove? Sassin, Anna
101 100 Searching for Sassin, Anna...
102 101 Removing Sassin, Anna!
103 102
104 103 Enter a command? 5
105 104
106 105 Who would you like to remove? Smith, Will
107 106
108 107 Searching for Smith, Will...
109 108 I'm sorry, "Smith, Will" was NOT found!
110 109
111 110 Enter a command? 6
112 111
113 112 CLEARING LIST:
114 113 Removing Belcher, Ura!
115 114 Removing Ding, Bill!
116 115 Removing Dwyer, Barb!
117 116 Removing Hogg, Ima!
118 117 Removing Lear, Shanda!
119 118 Removing Longbottom, Iva!
120 119 Removing Payne, Royal!
121 120
122 121 The list has been cleared!
123 122
124 123
125 124 Enter a command? 2
126 125
127 126 Can't display an empty list
128 127
129 128 Enter a command? 3
130 129
131 130 Yes, the list is empty.
132 131
133 132 Enter a command? 4
134 133
135 134 Can't search an empty list
136 135
137 136 Enter a command? 5
138 137
139 138 Can't remove from an empty list!
140 139
141 140 Enter a command? 6
142 141
143 142 The list has been cleared!
144 143
145 144 Enter a command? x
146 145 **** Please enter a NUMBER between 0 and 6 ****
147 146
```

```
148 Enter a command? 7
149 **** The number 7 is an invalid entry      ****
150 **** Please input a number between 0 and 6 ****
151
152 Enter a command? -1
153 **** The number -1 is an invalid entry      ****
154 **** Please input a number between 0 and 6 ****
155
156 Enter a command? 0
```

```
1  #ifndef HEADER_H_
2  #define HEADER_H_
3
4  #include <iostream> // cin, cout.
5  #include <string>   // string datatype variables.
6  #include <iomanip>  // fixed, setw, setprecision.
7  #include <fstream>
8  #include <limits>
9  #include <ios>
10 using namespace std;
11
12 enum Menu {
13     EXIT = 0,
14     CREATELIST,
15     DISPLIST,
16     ISEMPY,
17     SEARCH,
18     REMOVE,
19     CLEAR
20 };
21
22 struct PersonNode {
23     string name;
24     char gender;
25     int age;
26     PersonNode* next;
27     PersonNode* prev;
28 };
29
30 const int INPUT_COL = 14; // CALC - setw size for display column.
31
32 /*****
33  * CreateList
34  *   This function will create a list and take all the data from input file.
35  *   ==> returns nothing.
36  *
37  *   *****/
38 /
39 void CreateList(PersonNode*& head); // IN & CALC - List.
40
41 /*****
42  * Displist
43  *   This function will receive a list and display all of it.
44  *   ==> returns nothing.
45  *
46  *   *****/
47 /
48 void Displist(PersonNode* head);
49
```

```
46 /*****
47 * IsEmpty
48 *   This function will receive a list and check if it's empty.
49 *   ==> returns nothing.
50 *
51 void IsEmpty(PersonNode* head); // IN & CALC - Queue.
52
53 /*****
54 * DisplList
55 *   This function will receive and search for a person and displays their
56 *   information.
57 *   ==> returns nothing.
58 *
59 void Search(PersonNode* head);
60
61 /*****
62 * Remove
63 *   This function will receive a list and remove a person from it.
64 *   ==> returns nothing.
65 *
66 void Remove(PersonNode*& head);
67
68 /*****
69 * Size
70 *   This function will receive a stack and check it's size and display it.
71 *   ==> returns nothing.
72 *
73 void Size(PersonNode* head); // IN & CALC - Queue.
74
75 /*****
76 * Clear
77 *   This function will receive a list of nodes and clear it.
78 *   ==> returns nothing.
79 *
80 void Clear(PersonNode*& head);
81
82 /*****
83 * PrintHeaderFile
84 *   This function will output the header information
```

85 * ==> returns nothing.

86

*****/

87 void PrintHeaderFile(ostream& output, // IN - output datatype.

88 string asName, // IN - assignment name

89 int asNum, // IN - assignment number

90 string studentName, // IN - student's name

91 string classInfo, // IN - class that is being taken

92 char asType, // IN - assignment type

93 long long studentID); // IN - student ID

94

95 #endif

96

97

```

1  #include "Header.h"
2
3  /*****
4   * PrintHeaderFile
5   *   This function will output the header information
6
7   * _____
8   * PRE-CONDITIONS
9   *   The following parameters need to have a defined value prior to calling
10  *   the function
11  *       asName: The name of the assignment given in the course
12  *       asNum: The number of the assignment given in the course
13  *       studentName: The name of the student writing the code
14  *       classInfo: The course name, date, and time of the class
15  *       asType: Will either output as a lab or an assignment
16  *       studentID: The Identification Number of the student
17  *****/
18 void PrintHeaderFile(ostream& output,      // IN - output datatype.
19     string asName,      // IN - assignment name
20     int asNum,          // IN - assignment number
21     string studentName, // IN - student's name
22     string classInfo,   // IN - class that is being taken
23     char asType,        // IN - assignment type
24     long long studentID) // IN - student ID
25 {
26     output << left;
27     output << "*****\n";
28     output << "*   PROGRAMMED BY : " << studentName << endl;
29     output << "*   " << setw(14) << "STUDENT ID " << ": " << studentID << endl;
30     output << "*   " << setw(14) << "CLASS " << ": " << classInfo << endl;
31     output << "*   ";
32
33     // PROCESSING - This will adjust setws and format appropriately based
34     //               on if this is a lab 'L' or assignment
35
36     if (toupper(asType) == 'L')
37     {
38         output << "LAB #" << setw(9);
39     }
40     else
41     {
42         output << "ASSIGNMENT #" << setw(2);
43     }
44     output << asNum << ": " << asName << endl;
45     output << "*****";
46     output << right << endl;

```



```
47  
48     return;  
49 }
```

```
1 /
   *****
   ***
2 * AUTHOR      : Andrew Gharios
3 * STUDENT ID  : 1449366
4 * LAB 10      : Creating an ordered list.
5 * CLASS       : CS1B
6 * SECTION     : M-TH: 5-7:20p
7 * DUE DATE    : 7/15/21
8 *****
   */
9 #include "Header.h"
10
11 /
   *****
   ***
12 * Creating an ordered list
13 *-----
   -
14 * This program will provide a menu for the user to be able to manipulate a,
15 * ordered list. The user has the option to
16 *-----
   -
17 * INPUT:
18 * input : user menu selection.
19 *****
   */
20 int main()
21 {
22     /
   *****
   ***
23     * CONSTANTS
24     *-----
   -
25     * OUTPUT - USED FOR CLASS HEADING
26     *-----
   -
27     * PROGRAMMER : Programmer's Name
28     * CLASS      : Student's Course
29     * SECTION    : Class Days and Times
30     * LAB_NUM    : Lab Number (specific to this lab)
31     * LAB_NAME   : Title of the Lab
32     *****
   /
33
34     const string AS_NAME = "Creating an ordered list";
```

```
35     const int AS_NUM = 10;
36     const string STUDENT_NAME = "Andrew Gharios";
37     const string CLASS_INFO = "M-Th 5-7:20p";
38     const char AS_TYPE = 'L';
39     const long long STUDENT_ID = 1449366;
40
41     PersonNode* head; // IN & CALC - Stack front.
42     int input; // IN & CALC - menu input.
43     Menu menu; // CALC - Menu option.
44     bool invalid; // CALC - Validation for input.
45
46     head = NULL;
47
48     PrintHeaderFile(cout, AS_NAME, AS_NUM, STUDENT_NAME, CLASS_INFO, AS_TYPE, ↗
        STUDENT_ID);
49
50     cout << "STACK MENU:\n";
51     cout << "1 - Create List\n";
52     cout << "2 - Display List\n";
53     cout << "3 - Is the list empty?\n";
54     cout << "4 - Search by name\n";
55     cout << "5 - Remove Node\n";
56     cout << "6 - Clear List\n";
57     cout << "0 - to Exit\n";
58
59     do
60     {
61         do
62         {
63             invalid = false;
64             cout << "\nEnter a command? ";
65             if (!(cin >> input))
66             {
67                 cout << "**** Please enter a NUMBER between 0 and 6 ****\n";
68                 cin.clear();
69                 cin.ignore(numeric_limits<streamsize>::max(), '\n');
70                 invalid = true;
71             }
72             else if (input < 0 || input > 6)
73             {
74
75                 cout << "**** The number " << input << " is an invalid entry ↗
                    ****\n";
76                 cout << "**** Please input a number between 0 and 6 ****\n";
77                 invalid = true;
78             }
79
80         } while (invalid);
81     }
```

```
82
83     cin.ignore(numeric_limits<streamsize>::max(), '\n');
84
85     menu = Menu(input);
86
87     switch (menu)
88     {
89     case EXIT:
90         break;
91     case CREATELIST:
92         if (head == NULL)
93         {
94             CreateList(head);
95         }
96         else
97         {
98             cout << "\nThere's already a created list.\n";
99         }
100        break;
101    case DISPLIST:
102        DispList(head);
103        break;
104    case ISEMPY:
105        IsEmpty(head);
106        break;
107    case SEARCH:
108        Search(head);
109        break;
110    case REMOVE:
111        Remove(head);
112        break;
113    case CLEAR:
114        Clear(head);
115        break;
116    }
117    } while (menu != EXIT);
118
119    return 0;
120
121 }
```

```
1  #include "Header.h"
2
3  /*****
4  * CreateList
5  *   This function will create a list and take all the data from input file.
6  *
7  * INPUTS:
8  *   head : list.
9  *
10 * No outputs.
11 *
12 * *****/
13 /
14 void CreateList(PersonNode*& head) // IN & CALC - List.
15 {
16     PersonNode* perPtr;    // CALC - Searching pointer.
17     ifstream inFile;      // CALC - Input file variable.
18     PersonNode* searchPtr; // CALC - Search item.
19     bool found;           // CALC - If search item was found.
20
21     inFile.open("input.txt");
22
23     perPtr = NULL;
24     searchPtr = NULL;
25     perPtr = new PersonNode;
26     found = false;
27
28     while (inFile && perPtr != NULL)
29     {
30         getline(inFile, perPtr->name);
31         inFile.get(perPtr->gender);
32         inFile >> perPtr->age;
33         inFile.ignore(numeric_limits<streamsize>::max(), '\n');
34
35         cout << "\nAdding: " << perPtr->name;
36
37         if (head == NULL || head->name > perPtr->name)
38         {
39             perPtr->next = head;
40             perPtr->prev = NULL;
41             if (head != NULL)
42             {
43                 head->prev = perPtr;
44             }
45             head = perPtr;
46         }
47         else
48         {
49             // ... (rest of the code)
50         }
51     }
52 }
```

```
48     searchPtr = head;
49     while (searchPtr->next != NULL && !found)
50     {
51         if (searchPtr->next->name > perPtr->name)
52         {
53             found = true;
54         }
55         else
56         {
57             searchPtr = searchPtr->next;
58         }
59     }
60
61     found = false;
62     perPtr->next = searchPtr->next;
63     perPtr->prev = searchPtr;
64     if (searchPtr->next != NULL)
65     {
66         searchPtr->next->prev = perPtr;
67     }
68     searchPtr->next = perPtr;
69 }
70
71     perPtr = new PersonNode;
72 }
73 cout << endl;
74
75     perPtr = NULL;
76     delete perPtr;
77     inFile.close();
78
79 }
```

```

1  #include "Header.h"
2
3  /*****
4  * Display List
5  *   This function will receive a list and display all of it.
6  *
7  * INPUTS:
8  *   head : list.
9  *
10 * No outputs.
11 *
12 *****/
13 /
14 void DisplList(PersonNode* head) // IN & CALC - List.
15 {
16     const int NUM_COL = 5;
17     const int NAME_COL = 25;
18     const int GNDR_COL = 9;
19
20     PersonNode* perPtr; // CALC - Person pointer.
21     int count;          // CALC - number of people count.
22
23     count = 1;
24     perPtr = head;
25
26     if (perPtr == NULL)
27     {
28         cout << "\nCan't display an empty list\n";
29     }
30     else
31     {
32         cout << left;
33         cout << setw(NUM_COL) << " #";
34         cout << setw(NAME_COL) << "NAME" << setw(GNDR_COL) << " GENDER" << " AGE\n";
35
36         cout << setw(NUM_COL) << string(NUM_COL - 1, '-') << setw(NAME_COL) << string(NAME_COL - 1, '-');
37         cout << setw(GNDR_COL) << string(GNDR_COL - 1, '-') << string(5, '-') << endl;
38         cout << right;
39     }
40
41     while (perPtr != NULL)
42     {
43         cout << left;
44         cout << " " << setw(NUM_COL - 1) << count;
45         cout << setw(NAME_COL) << perPtr->name;

```

```
45         cout << "      " << setw(GNDR_COL - 2) << perPtr->gender;
46         cout << perPtr->age << endl;
47         cout << right;
48
49         perPtr = perPtr->next;
50         count++;
51     }
52 }
```



```

1  #include "Header.h"
2
3  /*****
4  * Search
5  *   This function will receive a list and search for a user chosen person then
6  *   display that person's information.
7  *
8  * INPUTS:
9  *   head : list.
10 *
11 * No outputs.
12 *
13 * *****/
14 /
15 void Search(PersonNode* head) // IN & CALC - List.
16 {
17     const int COL_SIZE = 9; // Setw size.
18
19     string searchItem;    // IN/CALC - Search item.
20     PersonNode* searchPtr; // CALC - Searching pointer.
21     bool found;           // CALC - If search item was found.
22
23     searchPtr = head;
24     found = false;
25
26     if (head != NULL)
27     {
28         cout << "\nWho would you like to search for? ";
29         getline(cin, searchItem);
30         cout << "\nSearching for " << searchItem << "..." << endl;
31
32         while (!found && searchPtr != NULL)
33         {
34             if (searchItem == searchPtr->name)
35             {
36                 found = true;
37                 cout << left;
38                 cout << setw(COL_SIZE + 1) << "\nName: " << searchPtr->name;
39                 cout << endl << setw(COL_SIZE) << "Gender: " << searchPtr->gender;
40                 cout << endl << setw(COL_SIZE) << "Age: " << searchPtr->age << endl;
41             }
42             else
43             {
44                 searchPtr = searchPtr->next;
45             }
46         }
47     }
48 }

```

```
46     if (!found)
47     {
48         cout << "I\'m sorry, \"" << searchItem << "\"" was NOT found!\n";
49     }
50     searchPtr = NULL;
51 }
52 else
53 {
54     cout << "\nCan't search an empty list\n";
55 }
56 }
```

```
1  #include "Header.h"
2
3  #include "Header.h"
4
5  /*****
6  * IsEmpty
7  *   This function will receive a list and check if it's empty or not.
8  *
9  * INPUTS:
10 *   head : list.
11 *
12 * No outputs.
13 *
14 * *****/
15 /
14 void IsEmpty(PersonNode* head) // IN & CALC - List.
15 {
16     if (head == NULL)
17     {
18         cout << "\nYes, the list is empty.\n";
19     }
20     else
21     {
22         cout << "\nNo, the list is NOT empty.\n";
23     }
24 }
```

```
1  #include "Header.h"
2
3  /*****
4  * Remove
5  *   This function will receive a list and remove a person based on user's input
6  *
7  * INPUTS:
8  *   head : list.
9  *
10 * No outputs.
11 *
12 * *****/
13 /
14 void Remove(PersonNode*& head) // IN & CALC - List.
15 {
16     PersonNode* searchPtr; // CALC - Searching pointer.
17     string searchItem;      // IN/CALC - Search item.
18     bool found;             // CALC - If search item was found.
19
20     searchPtr = NULL;
21     found = false;
22
23     if (head != NULL)
24     {
25         cout << "\nWho would you like to remove? ";
26         getline(cin, searchItem);
27         cout << "\nSearching for " << searchItem << "... \n";
28
29         searchPtr = head;
30
31         while (!found && searchPtr != NULL)
32         {
33             if (searchItem == searchPtr->name)
34             {
35                 found = true;
36                 if (searchPtr == head)
37                 {
38                     head = searchPtr->next;
39                     head->prev = NULL;
40                 }
41                 else if (searchPtr->next == NULL)
42                 {
43                     searchPtr->prev->next = NULL;
44                 }
45                 else
46                 {
47                     searchPtr->next->prev = searchPtr->prev;
48                     searchPtr->prev->next = searchPtr->next;
49                 }
50             }
51             searchPtr = searchPtr->next;
52         }
53     }
54 }
```

```
48
49         searchPtr = NULL;
50         delete searchPtr;
51
52         cout << "Removing " << searchItem << "!\n";
53     }
54     else
55     {
56         searchPtr = searchPtr->next;
57     }
58 }
59
60 if (!found)
61 {
62     cout << "I'm sorry, \"" << searchItem << "\" was NOT found!\n";
63 }
64 }
65 else
66 {
67     cout << "\nCan't remove from an empty list!\n";
68 }
69
70 }
```

```
1  #include "Header.h"
2
3  /*****
4  * Clear
5  *   This function will receive a list of nodes and clear it.
6  *
7  *   INPUTS:
8  *   head : list.
9  *
10 *   No outputs.
11 *
12 *   *****/
13 /
14 void Clear(PersonNode*& head) // IN & CALC - List.
15 {
16     PersonNode* searchPtr; // CALC    - Searching pointer.
17
18     searchPtr = head;
19
20     if (searchPtr == NULL)
21     {
22         cout << "\nThe list has been cleared!\n";
23     }
24     else
25     {
26         cout << "\nCLEARING LIST:\n";
27         while (searchPtr != NULL)
28         {
29             cout << "Removing " << searchPtr->name << "!\n";
30             head = searchPtr->next;
31             delete searchPtr;
32             searchPtr = head;
33         }
34         searchPtr = NULL;
35         cout << endl;
36         cout << "The list has been cleared!\n" << endl;
37     }
38 }
```