```
 1 ************************************************************
 2 *   PROGRAMMED BY : Andrew Gharios
 3 *   STUDENT ID    : 1449366
 4 *   CLASS         : M-Th 5-7:20p
 5 *   LAB #9        : Implementing a Stack
 6 ************************************************************
 7 STACK MENU:
 8 1 - Add a person (PUSH)
 9 2 - Remove a person (POP)
10 3 - Is the stack empty? (IsEmpty)
11 4 - Who is on top? (PEEK)
12 5 - How many people are there? (SIZE)
13 0 - to Exit
14 Enter a command? 1
15
16 Who would you like to add?
17 Enter Name:   John Smith
18 Enter Gender: M
19 Enter Age:    32
20
21 <resdisplay menu>
22 Enter a command? 1
23
24 Who would you like to add?
25 Enter Name:   Grace Hopper
26 Enter Gender: F
27 Enter Age:    69
28
29 <resdisplay menu>
30 Enter a command? 4
31
32 PEEKING at
33 Name:         Grace Hopper
34 Gender:       F
35 Age:          69
36
37 <resdisplay menu>
38 Enter a command? 5
39 There are 2 people on the stack.
40
41 <resdisplay menu>
42 Enter a command? 1
43
44 Who would you like to add?
45 Enter Name:   Ada Lovelace
46 Enter Gender: F
47 Enter Age:    52
48
49 <resdisplay menu>
```

```
Enter a command? 4

PEEKING at
Name:         Ada Lovelace
Gender:       F
Age:          52

<resdisplay menu>
Enter a command? 3
The stack is NOT empty.

<resdisplay menu>
Enter a command? 2
POPPING
Name:         Ada Lovelace
Gender:       F
Age:          52


<resdisplay menu>
Enter a command? 1

Who would you like to add?
Enter Name:   Charles Babbage
Enter Gender: M
Enter Age:    25

<resdisplay menu>
Enter a command? 4

PEEKING at
Name:         Charles Babbage
Gender:       M
Age:          25

<resdisplay menu>
Enter a command? 5
There are 3 people on the stack.

<resdisplay menu>
Enter a command? 2
POPPING
Name:         Charles Babbage
Gender:       M
Age:          25


<resdisplay menu>
Enter a command? 5
```

```
 99 There are 2 people on the stack.
100
101 <resdisplay menu>
102 Enter a command? 4
103
104 PEEKING at
105 Name:         Grace Hopper
106 Gender:       F
107 Age:          69
108
109 <resdisplay menu>
110 Enter a command? 2
111 POPPING
112 Name:         Grace Hopper
113 Gender:       F
114 Age:          69
115
116
117 <resdisplay menu>
118 Enter a command? 5
119 There is one person on the stack.
120
121 <resdisplay menu>
122 Enter a command? 3
123 The stack is NOT empty.
124
125 <resdisplay menu>
126 Enter a command? 4
127
128 PEEKING at
129 Name:         John Smith
130 Gender:       M
131 Age:          32
132
133 <resdisplay menu>
134 Enter a command? 2
135 POPPING
136 Name:         John Smith
137 Gender:       M
138 Age:          32
139
140 <resdisplay menu>
141 Enter a command? 5
142 Nobody is on the stack!
143
144 <resdisplay menu>
145 Enter a command? 2
146 Can't POP from an empty stack!
147
```

```
<resdisplay menu>
Enter a command? 3
Yes, the stack is empty.

<resdisplay menu>
Enter a command? 4
There is nobody to PEEK at!

<resdisplay menu>
Enter a command? -1
**** The number -1 is an invalid entry     *****
**** Please input a number between 0 and 5 *****

<resdisplay menu>
Enter a command? 6
**** The number 6 is an invalid entry     *****
**** Please input a number between 0 and 5 *****

<resdisplay menu>
Enter a command? abc
**** Please enter a NUMBER between 0 and 5 ****

<resdisplay menu>
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 <limits>
8  #include <ios>
9  using namespace std;
10
11 enum Menu {
12              EXIT = 0,
13              PUSH,
14              POP,
15              ISEMPTY,
16              PEEK,
17              SIZE
18            };
19
20 struct PersonNode {
21                      string name;
22                      char gender;
23                      int age;
24                      PersonNode* next;
25                    };
26
27 const int INPUT_COL = 14; // CALC - setw size for display column.
28
29 /*****************************************************************************
30 * PUSH
31 *   This function will receive a stack and add a person on top of it.
32 *  ==> returns stack after adding person on top.
33 *                                                                            ↵
     *****************************************************************************↵
     /
34 PersonNode* Push(PersonNode* head); // IN & CALC - Stack
35
36 /*****************************************************************************
37 * POP
38 *   This function will receive a stack and remove the person on top.
39 *  ==> returns stack after removing person on top.
40 *                                                                            ↵
     *****************************************************************************↵
     /
41 PersonNode* Pop(PersonNode* head); // IN & CALC - Stack
42
43 void IsEmpty(PersonNode* head); // IN & CALC - Stack.
44
45 /*****************************************************************************
```

```
46  * Peek
47  *   This function will receive a stack and peek at the name on top of the stack
48  *   and display it.
49  *  ==> returns nothing.
50  *                                                                              ↵
      ****************************************************************************** ↵
      /
51  void Peek(PersonNode* head); // IN & CALC - Stack.
52
53  /*****************************************************************************
54  * IsEmpty
55  *   This function will receive a stack and check if it's empty or not.
56  *  ==> returns nothing.
57  *                                                                              ↵
      ****************************************************************************** ↵
      /
58  void IsEmpty(PersonNode* head); // IN & CALC - Stack.
59
60  /*****************************************************************************
61  * Size
62  *   This function will receive a stack and check it's size and display it.
63  *  ==> returns nothing.
64  *                                                                              ↵
      ****************************************************************************** ↵
      /
65  void Size(PersonNode* head); // IN & CALC - Stack.
66
67  /*****************************************************************************
68   * PrintHeaderFile
69   *   This function will output the header information
70   *
71                                                                                ↵
    *****************************************************************************/
72  void PrintHeaderFile(ostream& output,      // IN  - output datatype.
73      string asName,        // IN  - assignment name
74      int asNum,             // IN  - assignment number
75      string studentName,   // IN  - student's name
76      string classInfo,     // IN  - class that is being taken
77      char asType,          // IN  - assignment type
78      long long studentID); // IN  - student ID
79
80  #endif
81
```

```
/*****************************************************************************
* AUTHOR     : Andrew Gharios
* STUDENT ID : 1449366
* LAB #9     : Implementing a Stack.
* CLASS      : CS1B
* SECTION    : M-TH: 5-7:20p
* DUE DATE   : 7/8/21
*****************************************************************************/
#include "Header.h"

/*****************************************************************************
* Implementing a Stack
*----------------------------------------------------------------------------
* This program will provide a menu for the user to be able to manipulate a
* stack. The user has the option to Push, Pop, Peek, check the Size, and check
* if the stack is empty.
*----------------------------------------------------------------------------
* 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
    *************************************************************************/

```

```
35      const string AS_NAME = "Implementing a Stack";
36      const int AS_NUM = 9;
37      const string STUDENT_NAME = "Andrew Gharios";
38      const string CLASS_INFO = "M-Th 5-7:20p";
39      const char AS_TYPE = 'L';
40      const long long STUDENT_ID = 1449366;
41
42      PersonNode* head;  // IN & CALC - Stack.
43      int  input;        // IN & CALC - menu input.
44      Menu menu;         // CALC      - Menu option.
45      bool invalid;      // CALC      - Validation for input.
46
47      head    = NULL;
48
49      PrintHeaderFile(cout, AS_NAME, AS_NUM, STUDENT_NAME, CLASS_INFO, AS_TYPE,
          STUDENT_ID);
50
51      cout << "STACK MENU:\n";
52      cout << "1 - Add a person (PUSH)\n";
53      cout << "2 - Remove a person (POP)\n";
54      cout << "3 - Is the stack empty? (IsEmpty)\n";
55      cout << "4 - Who is on top? (PEEK)\n";
56      cout << "5 - How many people are there? (SIZE)\n";
57      cout << "0 - to Exit\n";
58
59      do
60      {
61          do
62          {
63              invalid = false;
64              cout << "Enter a command? ";
65              if (!(cin >> input))
66              {
67                  cout << "**** Please enter a NUMBER between 0 and 5 ****\n";
68                  cin.clear();
69                  cin.ignore(numeric_limits<streamsize>::max(), '\n');
70                  invalid = true;
71              }
72              else if (input < 0 || input > 5)
73              {
74
75                  cout << "**** The number " << input << " is an invalid entry
                         *****\n";
76                  cout << "**** Please input a number between 0 and 5 *****\n";
77                  invalid = true;
78              }
79              if (invalid)
80              {
81                  cout << "\n<resdisplay menu>\n";
```

```
82              }
83
84
85          } while (invalid);
86
87          cin.ignore(numeric_limits<streamsize>::max(), '\n');
88
89          menu = Menu(input);
90
91          switch (menu)
92          {
93          case 0:
94              break;
95          case 1:
96              head = Push(head);
97              break;
98          case 2:
99              head = Pop(head);
100             break;
101         case 3:
102             IsEmpty(head);
103             break;
104         case 4:
105             Peek(head);
106             break;
107         case 5:
108             Size(head);
109             break;
110         }
111
112         if (input != 0)
113         {
114             cout << "\n<resdisplay menu>\n";
115         }
116
117     } while (menu != EXIT);
118
119     return 0;
120
121 }
```

```cpp
#include "Header.h"

/*************************************************************************
* PUSH
*   This function will receive a stack and add a person on top of it.
*  INPUTS:
*  head : Stack.
*
*  OUTPUTS:
*  head : Stack(with added person).
*
*************************************************************************/
PersonNode* Push(PersonNode* head) // IN & CALC - Stack.
{
    PersonNode* perPtr; // CALC - Pointer for manipulatin of stack.

    perPtr = new PersonNode;

    if (perPtr != NULL)
    {
        cout << left;
        cout << endl;
        cout << "Who would you like to add?" << endl;
        cout << setw(INPUT_COL) << "Enter Name:";
        getline(cin, (*perPtr).name);
        cout << setw(INPUT_COL) << "Enter Gender:";
        cin  >> perPtr->gender;
        cin.ignore(10000, '\n');
        cout << setw(INPUT_COL) << "Enter Age:";
        cin  >> perPtr->age;
        cin.ignore(10000, '\n');
        cout << right;

        perPtr->next = head;
        head   = perPtr;
    }
    perPtr = NULL;
    delete perPtr;
    return head;
}
```

```cpp
#include "Header.h"

/*****************************************************************************
* POP
*   This function will receive a stack and remove the person on top.
*
*  INPUTS:
*  head : Stack.
*
*  OUTPUTS:
*  head : Stack(with removed person on top)
*
*****************************************************************************/
PersonNode* Pop(PersonNode* head) // IN & CALC - Stack.
{
    PersonNode* perPtr; // CALC - Pointer for manipulatin of stack.

    perPtr = head;

    if (perPtr != NULL)
    {
        cout << left;
        cout << setw(INPUT_COL) << "POPPING"  << endl;
        cout << setw(INPUT_COL) << "Name: "   << perPtr->name   << endl;
        cout << setw(INPUT_COL) << "Gender: " << perPtr->gender << endl;
        cout << setw(INPUT_COL) << "Age: "    << perPtr->age    << endl;
        cout << endl;
        cout << right;

        head = perPtr->next;


        return head;
    }
    else
    {
        cout << "Can't POP from an empty stack!" << endl;
    }

    perPtr = NULL;

    return head;
}
```

```cpp
#include "Header.h"

/*****************************************************************************
* Peek
*   This function will receive a stack and peek at the name on top of the stack
*   and display it.
*
* INPUTS:
*  head : Stack.
*
* No outputs.
*
*****************************************************************************/
void Peek(PersonNode* head) // IN & CALC - Stack.
{

    if (head != NULL)
    {
        cout << left;
        cout << endl;
        cout << "PEEKING at\n";
        cout << setw(INPUT_COL) << "Name:"   << head->name << endl;
        cout << setw(INPUT_COL) << "Gender:" << head->gender << endl;
        cout << setw(INPUT_COL) << "Age:"    << head->age << endl;
        cout << right;
    }
    else
    {
        cout << "There is nobody to PEEK at!\n";
    }

}
```

```cpp
#include "Header.h"

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

```cpp
#include "Header.h"

/*************************************************************************
* Size
*   This function will receive a stack and check it's size and display it.
*  INPUTS:
*   head  : stack
*
* No outputs.
*
*************************************************************************/
void Size(PersonNode* head) // IN & CALC - Stack.
{
    PersonNode* perPtr;
    int         count;

    perPtr = head;
    count  = 0;

    while (perPtr != NULL)
    {
        count++;
        perPtr = perPtr->next;
    }

    if (count == 0)
    {
        cout << "Nobody is on the stack!\n";
    }
    else if (count == 1)
    {
        cout << "There is one person on the stack.\n";
    }
    else
    {
        cout << "There are " << count << " people on the stack.\n";
    }
}
```

```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;
```

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