Assignment #2

Topic(s):	C, Makefiles, Writing modular code, Stack ADT
Date assigned:	Monday, September 11, 2017
Date due:	Monday, September 25, 2017
Points:	25

The purpose of this assignment is to have you implement a Stack ADT using a static array of pointers and typeless dynamic memory. This way, you can push any datatype onto your stack; that is, your stack is generic!!!! How cool is that???

Specifically, for this assignment you will:

1) create an Eclipse project called **GenericStaticStack** using:
   a. an include file called **stk.h**, which is the stack interface
   b. a source file **stk.c** which is the stack implementation
   c. a source file **stkdriver.c** which is the stack test driver that uses asserts to thoroughly test your stack functions.
   d. a make file called **Makefile** that is used to build all object files and executables for the project

A copy of stk.h exists on zeus in **/home/CS300Public/2017/02Files**. You are to copy stk.h from zeus and implement each function prototype specified in stk.h in a file called stk.c. Do not modify stk.h in any way or you will lose significant points. Here is a simple driver that tests some of your stack functions.

```
10 #include <stdio.h>
11 #include <stdlib.h>
12
13 #include "../include/stk.h"
14
15 /*****************************************************************************
16  Function:    success
17
18  Description: print a success message
19
20  Parameters:  pszStr - the message to print
21
22  Returned:    none
23  *****************************************************************************/
24 static void success (char *pszStr)
25 {
26   printf ("SUCCESS: %s\n", pszStr);
27 }
28
29 /*****************************************************************************
30  Function:    failure
31
32  Description: print a failure message
33
34  Parameters:  pszStr - the message to print
35
36  Returned:    none
37  *****************************************************************************/
38 static void failure (char *pszStr)
39 {
40   printf ("FAILURE: %s\n", pszStr);
41 }
42
43 /*****************************************************************************
44  Function:    assert
45
46  Description: if the expression is true, assert success; otherwise, assert
47               failure
48
49  Parameters:  szStr - the message to print
50
51  Returned:    none
52  *****************************************************************************/
53 static void assert (bool bExpression, char *pTrue, char *pFalse)
54 {
55   if (bExpression)
56   {
57     success (pTrue);
58   }
59   else
60   {
61     failure (pFalse);
62   }
63 }
```

```
/*****************************************************************************
 Function:    main

 Description: test all the functionality of the stack

 Parameters:  none

 Returned:    Exit Status
 *****************************************************************************/

int main ()
{
  Stack sTheStack;

  puts ("Program Start\n");

  puts ("SUCCESS TESTS:");

  stkLoadErrorMessages ();
  success ("Loaded Error Messages");

  stkCreate (&sTheStack);
  success ("Stack Created");

  assert (stkSize (&sTheStack) == 0, "Stack size is 0",
                                     "Stack size is NOT 0");

  stkTerminate (&sTheStack);
  success ("Stack Terminated");

  puts ("\nProgram End");

  return EXIT_SUCCESS;
}
```

To successfully complete this assignment:

1. Implement each of the functions for stk.h one at a time in a file called stk.c. Test each function in a driver stkdriver.c. Create a Makefile for the project GenericStaticStack. This time, when you create a C Project, Select Makefile project (instead of Executable -> Hello World ANSI C Project) and Empty Project. A Makefile is a textfile created at the same level as the folders include, src, and bin. The Rational project is a great example. Please see me if you have questions regarding Rational.
2. Once you have implemented each function, you are to write a driver that extensively tests each of the functions in your program. Part of your grade will be based on how well your driver tests each and every function in stk.h. Note: The driver that I supplied you is not a good example of extensively testing each of your functions. Your driver must have many well thought out assert statements. You will also need loops to write a proper driver.

<u>Part A (Due: Monday, September 18, 2017)</u>

For this part of the assignment, you are to implement stkLoadErrorMessages, stkCreate, stkIsFull, stkIsEmpty, and stkSize. Your driver is to test each of these functions for correctness. Once you have completed this portion of the assignment, you are to submit your solution. To do so,

1) clean your project and then create a tarball called **cs300_2A_punetid.tar.gz** (that is "your"

punetid) that contains all files for correctly compiling your program on zeus. How to do this:
   a. at the level of GenericStaticStack (after cleaning your project), type the command: **tar czf cs300_2A_punetid.tar.gz** GenericStaticStack OR use **make tarball** if you have a proper tarball target set up in your Makefile
2) use scp to transfer the tarball to zeus for testing
3) once you are sure your program works on zeus from the command line, then submit your tarball as you did in assignment #1

Part B (Due: Monday, September 25, 2017)

Implement the rest of the functions for stk.c and write a driver that extensively tests each of your functions. I will write several test drivers to test your program when grading.

As in part A, create a tarball called **cs300_2B_punetid.tar.gz**, scp a copy to zeus for testing BEFORE submitting your final solution.

If you find any mistakes or you think there are discrepancies, please email me ASAP. I will check into your issue, fix as necessary, and email the entire class if changes are made.

Hints:

1) Many of you will find that setting up your project correctly in Eclipse (including a proper Makefile) will be difficult & frustrating. The sooner you get this set up, the better AND you can see me EARLY for questions.
2) The first function you need to implement is stkLoadErrorMessages. Then write a minimal stkdriver.c that calls stkLoadErrorMessages. Finally, write the Makefile and test. If you've written any more code than this and you have Makefile issues, I will not help you until you show me only the minimal code I just asked for.
3) Pointers may also be another level of frustration, so here is a little sample code that, if you understand the code, will help you with the programming portion of part B:
4) You will need to use memcpy as opposed to strcpy.

```
typedef struct String *StringPtr;
typedef struct String
{
  char data[32];
  int size;
} String;
…
String sTheString;
StringPtr psTheString = &sTheString;
psTheString->data[0] = '\0';  // Now sTheString is the null string
psTheString->size = 0;
strcpy (psTheString->data, "Hello CS300");
psTheString->size = strlen (psTheString->data);
puts (psTheString->data);
```

Error Processing

1. Check any parameter pointer for NULL and report an appropriate error message for a NULL pointer. For instance, if psStack is NULL in stkCreate, report STK_NO_CREATE_ERROR. Also, check for empty and full where appropriate.
2. I've tried to make the error messages descriptive in their use. Feel free to stop on by and ask questions regarding error handling.