Assignment #2

Topic(s):	C, Makefiles, Writing modular code, Stack ADT
Date assigned:	Friday, September 9, 2016
Date due:	Wednesday, September 21, 2016
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 a project called **GenericStaticStack** using:
   a. an include file called **sstk.h**, which is the stack interface
   b. a source file **sstk.c** which is the stack implementation
   c. a source file **sstkdriver.c** which is the stack driver
   d. a make file called **Makefile** that is used to build all object files and executables for the project

2) A copy of sstk.h is shown below and also exists on zeus in **/home/CS300Public/2016/02Files**. You are to copy sstk.h from zeus and implement each function prototype specified in sstk.h in a file called sstk.c. Do not modify sstk.h in any way or you will lose significant points.

```
/*****************************************************************
 File name:  sstk.h
 Author:     Computer Science, Pacific University
 Date:       9.9.16
 Class:      CS300
 Assignment: Static Generic Stack
 Purpose:    Interface for a static stack of generic elements
 *****************************************************************/

#ifndef SSTK_H_
#define SSTK_H_

#include <stdbool.h>
#include <stdlib.h>
#include <string.h>

//*****************************************************************
// Constants
//*****************************************************************
#define NUMBER_OF_ERRORS 6
#define MAX_ERROR_CHARS 64
#define NO_ERROR 0
#define ERROR_STACK_EMPTY 1
#define ERROR_STACK_FULL 2
#define ERROR_NO_STACK_CREATE 3
#define ERROR_NO_STACK_TERMINATE 4
#define ERROR_NO_STACK_MEMORY 5

#define MAX_STACK_ELEMENTS 100
#define EMPTY_STACK 0
```

```
31
32 //*****************************************************************************
33 // Error Messages
34 //*****************************************************************************
35 #define LOAD_ERRORS strcpy(gszErrors[NO_ERROR], "No Error.");\
36 strcpy(gszErrors[ERROR_STACK_EMPTY], "Error: Empty Stack.");\
37 strcpy(gszErrors[ERROR_STACK_FULL], "Error: Full Stack.");\
38 strcpy(gszErrors[ERROR_NO_STACK_CREATE], "Error: No Stack Create.");\
39 strcpy(gszErrors[ERROR_NO_STACK_TERMINATE], "Error: No Stack Terminate.");\
40 strcpy(gszErrors[ERROR_NO_STACK_MEMORY], "Error: No Stack Memory.");
41
42 //*****************************************************************************
43 // User-defined types
44 //*****************************************************************************
45 typedef struct Stack *StackPtr;
46 typedef struct Stack
47 {
48   void *data[MAX_STACK_ELEMENTS];
49   int top;
50 } Stack;
51
52 //*****************************************************************************
53 // Function prototypes
54 //*****************************************************************************
55 extern void stkLoadErrorMessages ();
56 extern void stkCreate (StackPtr psStack);
57 extern void stkTerminate (StackPtr psStack);
58 extern bool stkIsFull (const StackPtr psStack);
59 extern bool stkIsEmpty (const StackPtr psStack);
60 extern void stkPush (StackPtr psStack, void *buffer, int size);
61 extern void *stkPop (StackPtr psStack, void *buffer, int size);
62 extern void *stkPeek (const StackPtr psStack, void *buffer, int size);
63 extern int stkSize (const StackPtr psStack);
64
65 #endif /* SSTK_H_ */
```

Here is a simple driver that tests some of your stack functions.

```c
/*****************************************************************************
 File name:  sstkdriver.c
 Author:     Computer Science, Pacific University
 Date:       9.9.16
 Class:      CS300
 Assignment: Static Generic Stack
 Purpose:    Test driver for a static stack of generic elements
 *****************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include "../include/sstk.h"

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

 Description: test all the functionality of the stack

 Parameters:  none

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

int main ()
{
  Stack sTheStack;
  char ch = 'a',
       popValue;

  puts ("Program Start");

  stkCreate (&sTheStack);

  stkPush (&sTheStack, &ch, sizeof (char));

  while (!stkIsEmpty (&sTheStack))
  {
    stkPop (&sTheStack, &popValue, sizeof (char));
    printf ("Data = %c\n", popValue);
  }
  stkTerminate (&sTheStack);

  puts ("Program End");

  return EXIT_SUCCESS;
}
```

To successfully complete this portion of the assignment:

1. Implement each of the functions for sstk.h one at a time in a file called sstk.c. Test each function in a driver sstkdriver.c. Create a Makefile for the project GenericStaticStack.
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 listed above. Note: The driver that I supplied you is not

good for testing your project.

Part A (Due: Friday, September 16, 2016)

For this part of the assignment, you are to implement stkLoadErrorMessages, stkCreate, stkTerminate, 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) 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 GenericStackStack (after cleaning your project), type the command: tar czf cs300_2A_punetid.tar.gz GenericStaticStack
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: Wednesday, September 21, 2016)

Implement the rest of the functions for sstk.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**, move 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. In fact, I'm going to help you by giving you my Makefile in **/home/CS300Public/2016/02Files**
2) 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:
3) You will need to use memcpy

```
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);
```