Assignment #3

Topic(s):	Dynamic Memory, Valgrind, Reading from Files
Date assigned:	Monday, September 26, 2016
Date due:	Wednesday, October 5, 2016
Points:	30

You should now have a correctly working generic static stack. Now, we have decided to change from a static stack to a dynamic stack. If we do this to a user of our static stack, we must be able to do so without the user having to alter their code in any painful way. They really shouldn't even know we've changed the implementation!!!!

For this assignment you will:

1) create a project called **GenericDynamicStack** using:
   a. an include file called **dstk.h**, which is the stack interface
   b. a source file **dstk.c** which is the stack implementation
   c. a source file **dstkdriver.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 dstk.h is shown below and also exists on zeus in **/home/CS300Public/2016/03Files**. You are to copy dstk.h from zeus and implement each function prototype specified in dstk.h in a file called dstk.c. Do not modify dstk.h in any way or you will lose significant points.

```
/*****************************************************************************
 File name:  dstk.h
 Author:     Computer Science, Pacific University
 Date:       9.25.16
 Class:      CS300
 Assignment: Dynamic Generic Stack
 Purpose:    Header for a dynamic stack of generic elements
 *****************************************************************************/

#ifndef DSTK_H_
#define DSTK_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

//*****************************************************************************
// Error Messages
//*****************************************************************************
#define LOAD_ERRORS strcpy(gszErrors[NO_ERROR], "No Error.");\
strcpy(gszErrors[ERROR_STACK_EMPTY], "Error: Empty Stack.");\
strcpy(gszErrors[ERROR_STACK_FULL], "Error: Full Stack.");\
strcpy(gszErrors[ERROR_NO_STACK_CREATE], "Error: No Stack Create.");\
strcpy(gszErrors[ERROR_NO_STACK_TERMINATE], "Error: No Stack Terminate.");\
strcpy(gszErrors[ERROR_NO_STACK_MEMORY], "Error: No Stack Memory.");

```

```
42 //****************************************************************************
43 // User-defined types
44 //****************************************************************************
45
46 typedef struct Node *NodePtr;
47 typedef struct Node
48 {
49   void *pData;
50   NodePtr psNext;
51 } Node;
52
53 typedef struct Stack *StackPtr;
54 typedef struct Stack
55 {
56   NodePtr psTop;
57   int size;
58 } Stack;
59
60 //****************************************************************************
61 // Function prototypes
62 //****************************************************************************
63 extern void stkLoadErrorMessages ();
64 extern void stkCreate (StackPtr psStack);
65 extern void stkTerminate (StackPtr psStack);
66 extern bool stkIsFull (const StackPtr psStack);
67 extern bool stkIsEmpty (const StackPtr psStack);
68 extern void stkPush (StackPtr psStack, void *pBuffer, int size);
69 extern void *stkPop (StackPtr psStack, void *pBuffer, int size);
70 extern void *stkPeek (const StackPtr psStack, void *pBuffer, int size);
71 extern int stkSize (const StackPtr psStack);
72
73 #endif /* DSTK_H_ */
```

Here is a simple driver that tests some of your stack functions. Notice the only difference between sstkdriver.c and dstkdriver.c is line #12.

```
1  /*****************************************************************************
2   File name:  dstkdriver.c
3   Author:     Computer Science, Pacific University
4   Date:       9.26.16
5   Class:      CS300
6   Assignment: Dynamic Generic Stack
7   Purpose:    Test driver for a dynamic stack of generic elements
8   *****************************************************************************/
9
10 #include <stdio.h>
11 #include <stdlib.h>
12 #include "../include/dstk.h"
13
14 /*****************************************************************************
15  Function:    main
16
17  Description: test all the functionality of the stack
18
19  Parameters:  none
20
21  Returned:    Exit Status
22  *****************************************************************************/
23
24 int main ()
25 {
26   Stack sTheStack;
27   char ch = 'a',
28        popValue;
29
30   puts ("Program Start");
31
32   stkCreate (&sTheStack);
33
34   stkPush (&sTheStack, &ch, sizeof (char));
35
36   while (!stkIsEmpty (&sTheStack))
37   {
38     stkPop (&sTheStack, &popValue, sizeof (char));
39     printf ("Data = %c\n", popValue);
40   }
41   stkTerminate (&sTheStack);
42
43   puts ("Program End");
44
45   return EXIT_SUCCESS;
46 }
```

To successfully complete this assignment:

1. Create a new project called GenericDynamicStack. Make sure that you create a new C project->Makefile project->Empty Project->Linux GCC. Add the directories (src, include, bin, and datafiles).
2. Implement each of the functions for dstk.h one at a time in a file called dstk.c.
3. Create a Makefile for the project GenericDynamicStack. Watch the Makefile videos and look at the Makefile for GenericStaticStack to help you create the project.
4. Once you have implemented each function, you are to write a driver (dstkdriver.c) that reads single words from a data file called palindromes.txt found in a folder called datafiles. The driver will print the word followed by [palindrome] or [not palindrome]. The driver is mostly written. All you have to do is write the function isPalindrome correctly. You can write this driver right now and test it with your GenericStaticStack. Only one change will be necessary to get this working with GenericDynamicStack.
5. You are to submit a tarball called **cs300_3_PUNetID.tar.gz** that contains your Eclipse projects GenericDynamicStack and GenericStaticStack after extensive testing on zeus.
6. You must begin using Subversion. In particular, you must show me at least 5 commits of GenericDynamicStack by the time this assignment is due. Failure to do so will cost you 5 points. When you are ready, just show me the history of GenericDynamicStack that has at least 5 commits.
7. List how many hours your worked on the assignment in the header comments.

If you've done this assignment correctly, you should be able to take your static stack driver and use it as a driver for the dynamic stack by simply changing the include to be dstk.h instead of sstk.h in the driver. No other changes are necessary. Now how cool is that!!!! ☺

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.