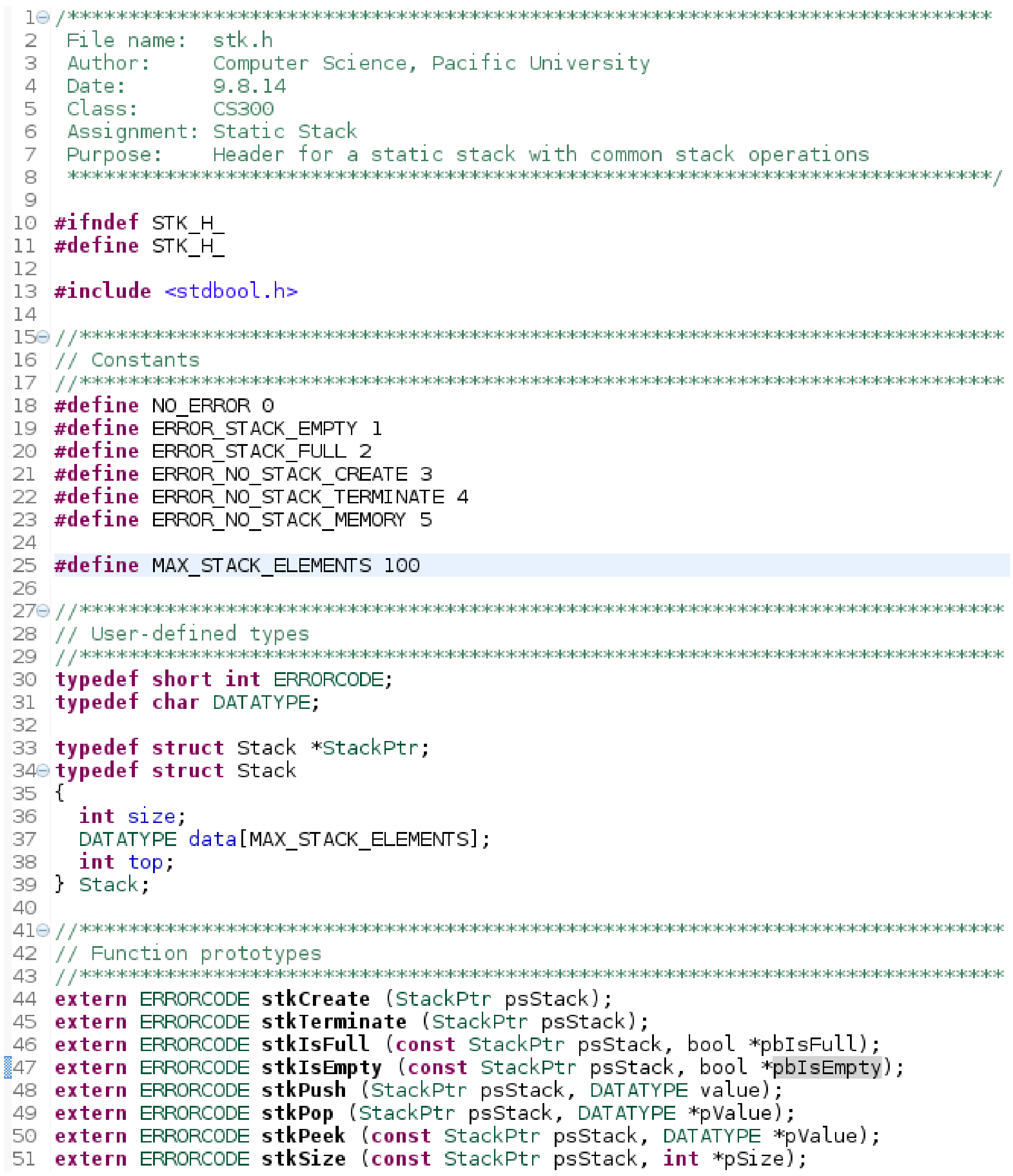
Assignment #2

Topic(s):	C, Makefiles, Writing modular code, Stack ADT, Multiple Projects
Date assigned:	Wednesday, September 10, 2014
Date due:	Wednesday, September 17, 2014
Points:	25

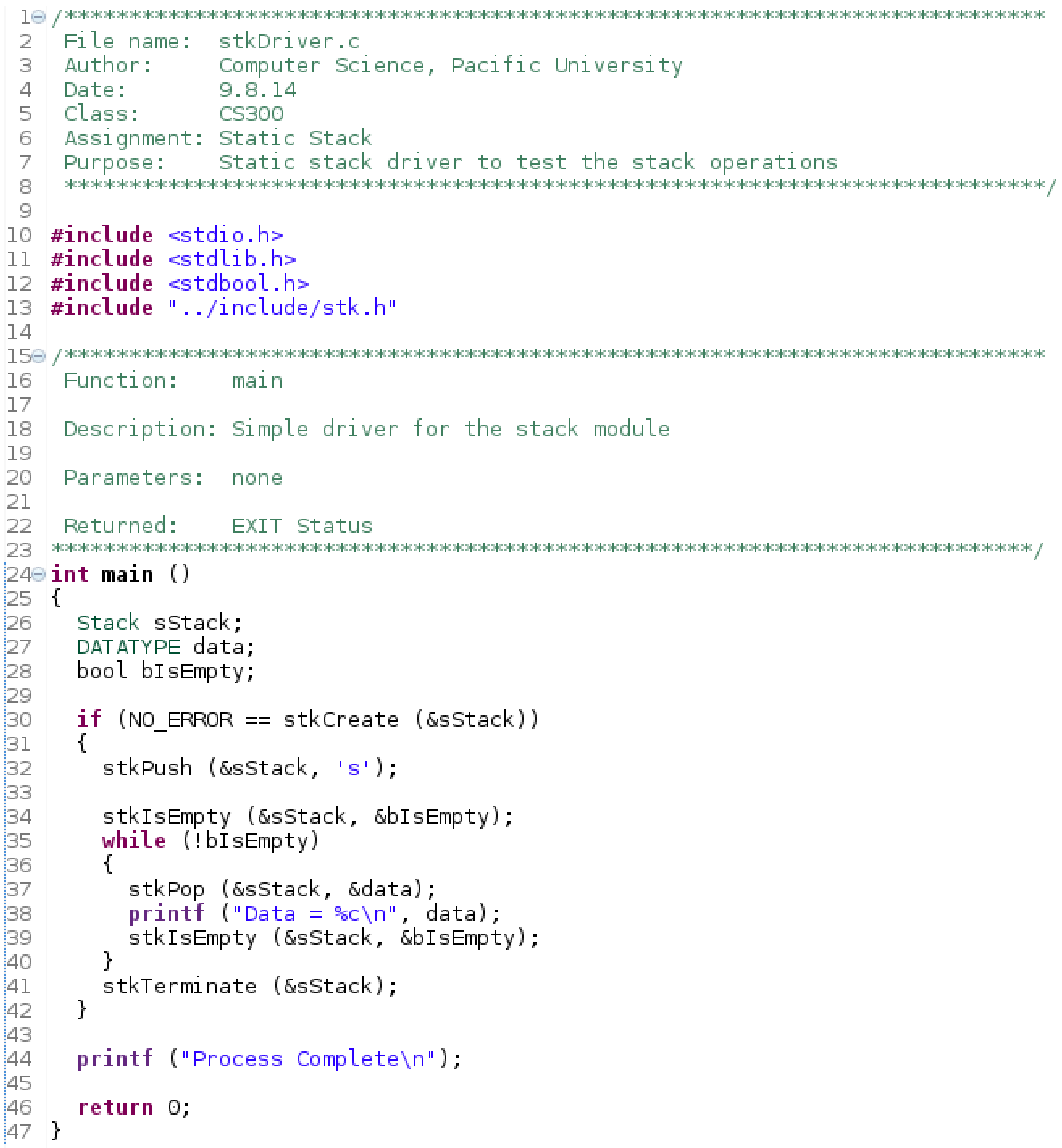
Part A

We have discussed the Stack ADT and looked at a static implementation of the Stack ADT. The purpose of this assignment is to have you implement the Stack ADT using static memory. Specifically, create a project called StaticStack with an include file, stk.h, and source files stk.c and stkdriver.c. A copy of stk.h is shown below and also exists on zeus in /home/CS300Public/2014/Assign2Files. You are to copy stk.h from zeus and implement each function prototype in stk.h in a file called stk.c. Do not modify stk.h in any way.



```
/**************************************************************************
 File name:  stk.h
 Author:     Computer Science, Pacific University
 Date:       9.8.14
 Class:      CS300
 Assignment: Static Stack
 Purpose:    Header for a static stack with common stack operations
 *************************************************************************/

#ifndef STK_H_
#define STK_H_

#include <stdbool.h>

//*************************************************************************
// Constants
//*************************************************************************
#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

//*************************************************************************
// User-defined types
//*************************************************************************
typedef short int ERRORCODE;
typedef char DATATYPE;

typedef struct Stack *StackPtr;
typedef struct Stack
{
  int size;
  DATATYPE data[MAX_STACK_ELEMENTS];
  int top;
} Stack;

//*************************************************************************
// Function prototypes
//*************************************************************************
extern ERRORCODE stkCreate (StackPtr psStack);
extern ERRORCODE stkTerminate (StackPtr psStack);
extern ERRORCODE stkIsFull (const StackPtr psStack, bool *pbIsFull);
extern ERRORCODE stkIsEmpty (const StackPtr psStack, bool *pbIsEmpty);
extern ERRORCODE stkPush (StackPtr psStack, DATATYPE value);
extern ERRORCODE stkPop (StackPtr psStack, DATATYPE *pValue);
extern ERRORCODE stkPeek (const StackPtr psStack, DATATYPE *pValue);
extern ERRORCODE stkSize (const StackPtr psStack, int *pSize);
```

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



```c
/*****************************************************************************
 File name:  stkDriver.c
 Author:     Computer Science, Pacific University
 Date:       9.8.14
 Class:      CS300
 Assignment: Static Stack
 Purpose:    Static stack driver to test the stack operations
 *****************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include "../include/stk.h"

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

 Description: Simple driver for the stack module

 Parameters:   none

 Returned:     EXIT Status
 *****************************************************************************/
int main ()
{
  Stack sStack;
  DATATYPE data;
  bool bIsEmpty;

  if (NO_ERROR == stkCreate (&sStack))
  {
    stkPush (&sStack, 's');

    stkIsEmpty (&sStack, &bIsEmpty);
    while (!bIsEmpty)
    {
      stkPop (&sStack, &data);
      printf ("Data = %c\n", data);
      stkIsEmpty (&sStack, &bIsEmpty);
    }
    stkTerminate (&sStack);
  }

  printf ("Process Complete\n");

  return 0;
}
```

To successfully complete this portion of the 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 StaticStack.
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 B

Now that you have a working implementation for the Stack ADT, you are to use this code to implement a palindrome checker. I have written all of the code in /home/CS300Public/2014/Assign2Files. All you have to do is put everything together to produce the executable.

To do this,

1. create a new project called PalindromeChecker
2. create an additional folder called testcases at the same level as bin and src
3. create a file called palindrome1.txt in testcases and copy the testcases from palindrome1.txt found in Assign2Files
4. create palindromeChecker.c in src and Makefile using the files from Assign2Files
5. build the project and run

If all goes well, you will see the following output:

```
PALINDROME CHECKER
---------- -------

mom [palindrome]
palindrome [not palindrome]
racecar [palindrome]
rotator [palindrome]
computer science [not palindrome]
Madam I'm Adam [palindrome]
```

Now it's time to submit your solution. You first must create a tarball called **cs300_2_punetid.tar.gz** that contains both Eclipse projects StaticStack and PalindromeChecker. At the level of both folders, the command is:

```
        tar czf cs300_2_punetid.tar.gz StaticStack PalindromeChecker
```

scp the file over to zeus, extract and test!!!!!

Once you are sure the tarball extracts correctly and works properly, submit the tarball as you did for assignment #1.

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