CSCI 4061
Introduction to Operating Systems

Assignment 1
This must be done individually.
Due Date: September 24, 2017

Instruction: This assignment contains 2 parts. Solution of part 1 must be saved in a PDF file and put in the folder named 'Theory'. Solution of part 2 must be put in the folder 'Project'. These 2 folders must be zipped (.zip or .tar or.tar.gz) in a single file and submitted using the Homework Submission link on the moodle web page.

Part - 1 Theory (20 points)

Question 1 (5 points): Explain what is the buffer overflow problem.

Question 2 (5 points): What are the important functions of an OS kernel?

Question 3 (5 points): Identify the steps that are performed by the shell when user executes 'ls' command.

Question 4 (5 points): Which of the instructions should be allowed only in the privileged (kernel) mode?

a) Disable all interrupts.

b) Set the time-of-the day clock.

c) Set a timer.

d) Read from kernel memory.

e) Fetch an instruction from kernel memory.

f) Read the time-of-the-day clock.

g) Change the base-bound register values.

h) Turn off timer interrupt.

i) Write to kernel memory.

j) Change the program counter.

Part - 2 Programming Project (100 points)

Objectives:
The objective of this assignment is to learn the basic concepts in shell programming using TC Shell or Bash.

Assignment Statement:

You can do this assignment using **either** tcsh **or** bash.

In this assignment you are required to write a shell program that will provide several useful functions for you. These functions include:

1. Searching for a filename pattern in a given directory and all of its subdirectories.
2. Calculating the total of the size of all files contained in a directory and its subdirectories.
3. Finding all zero length files contained in a directory and its subdirectories.
4. Creating backup files for all files specified using a filename pattern and contained in a given directory and its subdirectories.

Your program will prompt the user to make a selection from a menu by responding with a single character. A sample menu is shown below. (Please note that the examples shown are for illustrative purposes; your interface may differ, but should be user-friendly and informative.)

You will find skeleton programs (assignment1_skeleton.bash and assignment1_skeleton.tcsh) from the assignment 1 section on the moodle web page. You will need to fill the code for the following four functions.

Example:

SELECT THE FUNCTION YOU WANT TO EXECUTE:
1. Search for files
2. Calculate total length
3. Find zero length files
4. Create backup files
ENTER YOUR CHOICE:

(The user should enter either 1, 2, 3, or 4.)
For each of these cases, your program should prompt the user for inputs specific to that function.

Case 1:
For the file search option, your program should prompt the user for a directory and a file name.

Example:

EXECUTING FILE SEARCH FUNCTION
ENTER DIRECTORY NAME:
ENTER FILE NAME:

- Your program should do error checking to make sure that the directory name is valid.
- For each file matching the specified name, your program should print the full pathname for the file and print the information provided by the command "ls -l" for an ordinary file and "ls -ld" for a directory file.

Case 2:

Example:

Your program should output the total size of all files in that directory and recursively in all of its subdirectories:

Please note that you are NOT allowed to use the "du" command in writing the code for this part.

Example:

The sum of the size of all files in this directory is:

Case 3:

Example:

EXECUTING SEARCH FOR ZERO LENGTH FILES
ENTER DIRECTORY NAME:

- For each zero-length file, your program should print the full pathname for the file only in that directory (do not worry about recursive directory hierarchy at this stage).

Case 4:

Example:

EXECUTING FILE BACKUP FUNCTION
ENTER DIRECTORY NAME:
ENTER FILE NAME PATTERN:

The user should give a directory name and a file name. Your program should take the following actions:

- For each ordinary file found in the directory (and its sub-directories) specified by the user, your program should check if file-name.bak exists. If no file with ".bak" extension exists, it will create a copy of the file with extension ".bak".
    - If a file with ".bak" extension exists, your program will compare the file with the backup file using "diff". If "diff" generates zero-length output, that means that there is no need to create a new backup
    - If "diff" generates a non-zero length output, that means the files differ.
    - In this case your program should move the backup file to a file with extension ".bak<today's date>" in the form filename.bak-MM-DD-YYYY(e.g. filename.bak-01-23-2013), and create a copy of the matching file with extension ".bak". You can use date command with the following option to get the desired date string: `date +%m-%d-%Y`
    - Your program should print the full pathname for each file for which a backup is created
    - It should also report if an existing backup was found to be the same as a matching file and no backup was created
    - It should also report if an existing backup was moved to a new name with today's date appended to it.
- For a directory file matching the given file name, your program will create a new directory with name filename.bak by recursively copying all files and directories in that new directory. The files and the sub-directories would have their original names.
    - If a directory with filename.bak already exists, your program will move that to "filename.bak<today's date>" (with the date in the format as noted above) and create a new backup directory with name filename.bak.
    - If a backup for a directory is created, your program will report it by printing the directory's pathname.
    - Your program will report if an existing backup directory was found and moved to ".bak<today's date>".

Things to Submit:

- Please include the required information according to the guidelines on your submission and indicate the operating system (Linux, Unix, Mac OS) that was used by you for testing your code.
- Include the following header information in your code.
- Name your shell-script code file as either assignment1.tcsh or assignment1.bash, depending on the shell your code uses.

```
# CSci 4061 Fall 2017
# Assignment# 1
# student name:
# student id:
# x500 id:
# Operating system on which you tested your code: Linux, Unix, Mac OS
# CSELABS machine: <machine you tested on e.g.: xyz.cselabs.umn.edu>
```

Note on Comments:

Please comment your code. It is a good practice to comment your code so that it is understandable to you and others. Proper commenting will save you lot of time while debugging.

Also see to it that you do not have excessive comments. Comments should be crisp clear and to the point. We reserve the right to deduct up to 10% of the allocated points if there are NO comments in the code.

Grading guidelines:

Grading would be broken up into the following main parts:

Case 1: Search for files	20%
Case 2: Calculate total length	20%
Case 3: Find zero length files	20%
Case 4: Create backup files	30%
Documentation and readability: Includes summary of the programs functionality, instructions if any for using the program, general assumptions in the program, corner cases in which the program would not work, and any other general comments on main assignment page.	10%