CSC3320 System Level Programming
Homework 4

Due at 11:59 pm on Oct. 28th, 2014

Part1: Getting Started with shell script. (5 points)
To create a shell script for execution in Unix system, you need to follow three steps:
(1) Create a shell script file with cat or vi.
  e.g. $vi GetStarted.sh

(2) Type your script in the editor and remember that the first line should declare which kind of shell this script is based on.
  e.g. #!/bin/bash

(3) Save the file and change permission type so it can be executable.
  e.g. $chmod 777 GetStarted.sh

(4) In the current directory, execute the shell script.
  e.g. $./GetStarted.sh

Now write a bash script (GetStarted.sh) to accept your first name as an argument and display the success information. The output could be like this. Also attach a screenshot of your output.

$/GetStarted.sh yuan
Congratulations, yuan! You already known how to get started with shell script.

(Hint: to get the value of argument like “yuan” in the above example, you can use $1)
Part2: (10 points)
Here is the definition of Fibonacci numbers (from Wikipedia):

In mathematics, the Fibonacci numbers or Fibonacci series or Fibonacci sequence are the numbers in the following integer sequence:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...

By definition, the first two numbers in the Fibonacci sequence are 0 and 1, and each subsequent number is the sum of the previous two.

In mathematical terms, the sequence \( F_n \) of Fibonacci numbers is defined by the recurrence relation

\[
F_n = F_{n-1} + F_{n-2},
\]

with seed values

\[
F_0 = 0, \ F_1 = 1.
\]

Write a bash script (fib\_lastname.sh) calculating Fibonacci number. It takes one argument \( n \) (\( n \geq 0 \), do not try numbers > 13), which may not work correctly due to number range limitation. And attach a screenshot of your output. The output looks like:

```
$./fib.sh 8
fib 8 = 21

$ ./fib.sh 10
fib 10=55
```

Submission:

- Upload an electronic copy (MS word or pdf) of your answer sheet (including the content in your shell scripts and screenshots of outputs) to the folder named “HW4” of the dropbox in the desire2learn system.
- Name your file in the format of HW4\_FisrtnameLastname (eg. HW4\_YuanLong.docx, HW4\_YuanLong.pdf)
- Please add the homework number and your name at the top of your answer sheet.
- Upload the two shell script files “fib\_lastname.sh” and “GetStarted.sh” to the desire2learn system. And please write your name as a comment at the second line in the file.
Hint:

For details about return value and recursion of function in bash, please follow two sections including good examples that you may start with.

● Returning from a Function

The return command returns the flow of control back to the caller, and has the following syntax:

```
return [value]
```

When return is used without an argument, the function call returns immediately with the exit code of the last command that was executed in the function; otherwise, it returns with its exit code set to value. If a return command is executed from the main script, it’s equivalent to an exit command. The exit code is accessible from the caller via the `?` variable. Here’s an example function that multiplies its arguments and returns the result:

```
$ cat func3.sh
f ()
#
  two-parameter function.
{
  (( returnValue = $1 * $2 ))
  return $returnValue
}
#
  main program.
f 3 4
result=?
# save exit code.
result=$result
# echo return value from function was $result

$ sh func3.sh
return value from function was 12
$_
```
Recursion

With careful thought, it's perfectly possible to write recursive functions. One approach uses the exit code to return the result and the second uses standard output to echo the result.

E.g. Recursive Factorial, using Exit Code

```bash
factorial () # one-parameter function
{
    if (( $1 <= 1 ))
        return 1 # return result in exit code.
    else
        ( ( tmp = $1 - 1 ))
        factorial $tmp
        ( ( result = $? * $1 ))
        return $result # return result in exit code.
    fi
}
# main program.
factorial 5 # call function
echo factorial 5 = $? # display exit code.
```