Final Exam: CS 35L Section 7

University of California, Los Angeles

Fall 2017

Name: ______________________________

Student ID: ______________________________

Question	Points
1	/12
2	/16
3	/16
4	/16
5	/20
6	/20
Total	

Instructions:

1. This examination contains 12 pages in total, including this page.
2. You have **three (3) hours** to complete the examination.
3. Write your answers in this exam paper. The draft papers will not be accepted.
4. You may use any printed resources, including lecture notes, books, slides, assignments. **You cannot use any electronics.**
5. Please finish the final **independently**. If you have any question, please raise your hand.
6. There will be partial credit for each question.
7. Best of luck!

Question 1: Concept Question

[12 pts] Choose **two (2)** of the following three questions and answer them concisely.

(a) What is cloud computing? What is the advantage (list at least two) of it?

(b) What are the different levels (distance to hardware) of programming languages? Why do we need each of them?

(c) Draw a simple diagram of how public-key encryption system works (what are the keys involved, how to use keys for encryption/decryption).

Question 2: Unix Command

[16 pts]

The following is the subset of the man page for the "df" command.

```
DF(1)                     BSD General Commands Manual                    DF(1)

NAME
     df -- display free disk space

SYNOPSIS
     df [-b | -h | -H | -k | -m | -g | -P] [-ailn] [-t] [-T type]
        [file | filesystem ...]

LEGACY SYNOPSIS
     df [-b | -h | -H | -k | -m | -P] [-ailn] [-t type] [-T type] [file |
     filesystem ...]

DESCRIPTION
     The df utility displays statistics about the amount of free disk space
     on the specified filesystem or on the filesystem of which file is a
     part.  Values are displayed in 512-byte per block counts.  If neither
     a file or a filesystem operand is specified, statistics for all
     mounted filesystems are displayed (subject to the -t option below).

     The following options are available:

     -a      Show all mount points, including those that were mounted with
             the MNT_IGNORE flag.

     -b      Use (the default) 512-byte blocks.  This is only useful as a
             way to override an BLOCKSIZE specification from the environ-
             ment.

     -g      Use 1073741824-byte (1-Gbyte) blocks rather than the default.
             Note that this overrides the BLOCKSIZE specification from the
             environment.

     -i      Include statistics on the number of free inodes. This option
             is now the default to conform to Version 3 of the Single UNIX
             Specification (``SUSv3'') Use -P to suppress this output.

     -k      Use 1024-byte (1-Kbyte) blocks, rather than the default.  Note
             that this overrides the BLOCKSIZE specification from the envi-
             ronment.

     -l      Only display information about locally-mounted filesystems.

     -m      Use 1048576-byte (1-Mbyte) blocks rather than the default.

     -n      Print out the previously obtained statistics from the filesys-
             tems.  This option should be used if it is possible that one
             or more filesystems are in a state such that they will not be
             able to provide statistics without a long delay.  When this
```

```
         option is specified, df will not request new statistics from
         the filesystems, but will respond with the possibly stale sta-
         tistics that were previously obtained.

-P       Use (the default) 512-byte blocks.  This is only useful as a
         way to override an BLOCKSIZE specification from the environ-
         ment.

-T       Only print out statistics for filesystems of the specified
         types.  More than one type may be specified in a comma sepa-
         rated list.  The list of filesystem types can be prefixed with
         ``no'' to specify the filesystem types for which action should
         not be taken.  For example, the df command:

               df -T nonfs,mfs

         lists all filesystems except those of type NFS and MFS.  The
         lsvfs(1) command can be used to find out the types of filesys-
         tems that are available on the system.
```

Given the above instructions, you should specify the sequence of commands you use to finish the following:

(a) Use df to check the free disk space on the computer. The results should only include the locally-mounted filesystems, and exclude the autofs type filesystems. The display should use 1-Gbytes blocks.

(b) Use pipeline and find all the lines that include the keyword "**100%**" in the previous output. Save the extracted results as df.txt to your HOME directory.

(c) Go to your HOME directory. For df.txt, grant r/w/x permission to the owner, r/x permission to the group, and r permission to anyone.

(d) Change the filename to df_new.txt, create a new directory called test under HOME directory, and copy the file into this directory.

Question 3: Bash Programming

[16 pts] Write a bash script "easy" to finish the following tasks:

(a) The script takes two command line arguments (i.e. one extra parameter except for your script name). The second parameter is the path of a directory.

(b) This script will check whether the input parameter is in the $PATH variable. If not, add this directory into the $PATH variable.

(c) Under this directory, find the oldest regular file and gets the length of its filename. Denote this length as x.

(d) Under this directory, find the number of files that is two-character long. Denote this number as y.

(e) Write a function which calculates $x^3 + y^2 + x \cdot y$. Call this function and print the result to the standard output.

(f) Your script should handle two exceptions and outputs "error": when the input parameter number does not satisfy the requirement, and when the second argument is not a valid directory.

Question 4: Python Programming

[16 pts] In this question, you shall write a Python script to calculate the Euler's totient function of an integer n. Do not be scared away from the question; It is easy after you read and understand the following explanation. Hope that you also learn something from this question.

In number theory, Euler's totient function counts the positive integers up to a given integer n that are relatively prime (or co-prime)[1] to n, which is denoted as $\varphi(n)$. As an example, $\varphi(9) = 6$ since $1, 2, 4, 5, 7, 8$ are relatively prime to 9, while $3, 6, 9$ are not. $\varphi(10) = 4$ since $1, 3, 7, 9$ are relatively prime to 10, while $2, 4, 5, 6, 8, 10$ are not.

There exists an efficient way to calculate $\varphi(n)$. The critical step is to calculate prime factorization of a given n, which means that we will express an integer as the multiplication of a set of prime integers. For example, $360 = 2\times2\times2\times3\times3\times5 = 2^3\times3^2\times5^1$. After we get this factorization, it is easy to calculate the $\varphi(n)$. Suppose that $n = p_1^{k_1}\cdots p_r^{k_r}$, where $p_1 \cdots p_r$ are different prime numbers, $\varphi(n) = n(1-1/p_1)(1-1/p_2)\cdots(1-1/p_r)$. For example, $10 = 2^1 \times 5^1$, and $\varphi(10) = 10 \times (1-1/2) \times (1-1/5) = 4$; $360 = 2^3 \times 3^2 \times 5^1$, and $\varphi(360) = 360 \times (1-1/2) \times (1-1/3) \times (1-1/5) = 96$.

Luckily enough, you are not asked to calculate the prime factorization of n. Instead, there exists an library called zhaowei that helps you do it. In this library, you can call a function called "p-factor", which takes an integer as the input, and outputs a dictionary which represents the prime factorization of the input.

Calls p-factor(10) returns a dictionary {2:1, 5:1}, since $10 = 2^1 \times 5^1$.
Calls p-factor(360) returns a dictionary {2:3, 3:2, 5:1}, since $360 = 2^3 \times 3^2 \times 5^1$.

Note that if the input for p-factor() is not an integer greater than 1, the output is an empty dictionary.

(a) Describe how to install the package zhaowei using Python package management tools.

(b) Write a Python 2 scripts that takes one parameter n from the standard input. Calculate and print the Euler's totient function $\varphi(n)$ of n. You shall utilize the library zhaowei and the function p-factor.

(c) Your script should output "error" if the input is not a positive integer, and $\varphi(n)$ otherwise. You **do not** have to consider any other exceptions. Hint: $\varphi(1) = 1$ should be considered separately because p-factor only works for integer greater than 1.

[1] Two integers x and y are said to be co-prime if the only positive integer that divides both of them is 1. E.g., 10 and 9 are co-prime, while 10 and 8 are not. As a side note, obviously, any two prime numbers are co-prime.

Question 5: C Programming: Multithreading, Git

[20 pts] There exists a Git directory in the remote Git server. `https://github.com/ZhaoweiTan/final.git`. This Git project only has one branch master. Inside this directory is a single file called vector.c. Inside the file it reads:

```
double dot_product(double v[], double u[], int n) {
    int i;
    double result = 0;
    for (i = 0; i < n; i+=1)
    {
      result += v[i]*u[i];
    }
    return result;
}
```

You are asked to do the following. Your answer should include the commands you use and the program you write.

(a) Git clone the code to your local computer under $HOME/test directory. (Suppose the directory is already there, but you have to change directory first)

(b) Create a new branch called "test" and switch to it.

(c) In this new branch, create a new file called mt_vector.c.

(d) Understand the code in vector.c; write a multithreading version of it in mt_vector.c. There should be at least a function called dot_product_multithread in your program, which utilizes **four (4)** threads and returns the identical results. When an error occurs when handling pthread, prints "error" and returns 999.

(e) Commit the changes and merge the changes to the master branch.

Question 6: C Programming: System Call and Compiling

[20 pts]

(a) Firstly, please list the four compiling stages, and explain what each step roughly does.

(b) Suppose that you have a file called input.txt under your HOME directory. Write a main.c script. In the main function use system calls to read the content in this file.

(c) Suppose that the content in input.txt is a string. Write a C file called reverse.c and its header reverse.h. In these files write a function reverse_string that reverses a string.

(d) In main.c, call the function in reverse.c on the string you just read from input.txt. You should dynamically load reverse_string function in reverse.c using dlopen and dlsym. You are supposed to close up in the end as well.

(e) Use system call to write the reversed string to the standard output.

(f) Your program should consider all the linking errors and system call errors. If an error occurs, simply returns -1. Otherwise, a successful main should return 0.

(g) Write a simple makefile that compiles your library and program.

(h) What command can check the dynamic libraries your program uses?