**Hint file for TMA02, 2018**

Note:

1. Read the question carefully and make sure you understand what the question asks. Never answer with a lot of un-related description. Unrelated answer doesn’t get any mark. On the contrary, marks may be deducted

2. Read carefully about submission format stated in “TMA page” of my teaching website.

3. Don’t include any of your source code, except for q1b, in your answer file (\*.doc).

**Q1(a)** This question is a gift to you.

Please refer to

* <http://www.cs.cf.ac.uk/Dave/C/node3.html>

and/or

* P.5 to 15 of my “s212\_2018\_Surgery5\_C\_programming.ppt”.

Don’t be too detail for each term. Provide key points for each term.

**Q1(b)** This question tests your understanding on syntax of C programming.

You can copy & paste the original code in \*.doc and then identify the bugs (at least 5) and describe briefly how to correct the bugs in \*.doc file.

* For compilation of C source code, please refer to P.11 to 15 of my “s212\_2018\_Surgery5\_C\_programming.ppt”.
* To debug:
  + 1. In labsupport,
       1. Create directory ~/tma2. In directory ~/tma2, create and edit the content to an ascii file ConsoleToFile.c exactly the same as in the listing .
       2. Compile the file using “gcc” and name executable file as ConsoleTofile.
       3. Read the error message carefully. Correct ConsoleToFile.c for any line (also 1 or 2 lines above or below) in-error.
       4. Recompile and then check the error message again. Repeat until no error in compilation.
       5. Set appropriate permission right to your compiled binary file, ConsoleToFile, so that it is executable.

**Please also include source code file “**ConsoleToFile.c” **in your answer file (\*.doc) so that I can comment on it. Locate the errors and explain how they should be corrected.**

**Q2a**

A. Update of the lab book: **P.120, Listing 2.2.4,** **following 2 lines should be added to line 60** of the listing:

**return 0;**

**}**

**B. Don’t include the numbers in the left of Listings 2.2.1, 2.2.2, 2.2.3, 2.2.4, 2.2.5,f 2.2.6 as source code. These numbers are shown for convenience of explaining the programs.**

**C. Set appropriate permission rights as below:**

1. **For directories:**

**~/public\_html/ 701**

**~/public\_html/cgi-bin/ 701**

1. **For html files:**

**~/public\_html/\*.html 704**

1. **For cgi programs:**

**~/public\_html/cgi-bin/\*.cgi 705**

**D. Relationship of programs:**

fibonacci

(Standalone program,

No input. Display first 20 Fibonacci numbers. 20 is hard-coded); Listing 2.2.1

(cgi program to accept single input n. Then display first n Fibonacci numbers in your browser); Listing 2.2.4

(form to accept single input n. Identical to program2.html);

Listing 2.2.6

(cgi program to accept single input n, where n is the number of Fibonacci numbers to display in browser.

It also displays counter s in browser, where s is the number of times this cgi program has been accessed. The value of counter is stored in a text file in web server This file is created by fibonacci3.cgi when fibonacci3.cgi is accessed at the first time. fibonacci3.cgi needs to increment counter every time it is accessed.); Listing 2.2.5

+

fibonacci3.cgi

program3.html

fibonacci.cgi

(cgi program directly accessed through browser.

Display first 20 Fibonacci numbers in your browser. 20 is hard-coded); Listing 2.2.2

(The number of Fibonacci numbers to display is NOT hard-coded. This form is used to accept a single input n, where n is the number of Fibonacci numbers to display in browser. n is 1 to 99); Listing 2.2.3

+

fibonacci2.cgi

program2.html

**E Submission:** Don’t include any source code in your \*.doc file. I will check your labsupport account.

**Q2b (don’t include source code for Q5, Q6a, Q6b in \*.doc. Just provide their location in your labsupport account)**

**q1.** **Please refer to P.36 to 38 of study unit 7**

**q2. Please refer to P.34 to 35 of study unit 7**

**char input[9];**

q3. Think about:

* + For program2.html, what is the name of the input to the form? What is the size of the input name?
  + For program2.html, what is the maximum length of the input value to the form?
  + For the string read by getInput() from stdin, what is its format and size? Refer to item 3 of my CGI workshop
  + Usually string is ended with NULL character.

**return atoi(input + 6);**

q4. Think about:

* + atoi() is to convert a “numeric string” to a integer value.
  + Input to atoi() is a pointer to a string.
  + Think about pointer concept by following example:

char string[20]=”12345678901234”;

char \*p;

p=&string[0];

Then (p+1) is also a pointer and \*(p + 1) = ”2345678901234”,

Then (p+2) is also a pointer and \*(p + 2) = ”345678901234”,

Then (p+6) is also a pointer and \*(p + 6) = ”78901234”,

What is the value of atoi(p+6)?

q5.

(Form to accept multiple inputs. Length of each input can be from 1 to 10. One of the input is n, where n is the number of Fibonacci numbers to display in browser. Value of n can be 1 to 99. The other inputs can be any you wanted.)

program4.html

+

(cgi program to accept a CGI input string.

1. Modify getInput() in fibonacci3.cgi to getInput(arg1)

2. Modify part of main() in fibonacci3.cgi so that getInput(arg1) is called with a input parameter. )

fibonacci4.cgi

getInput(arg1) should be able to do followings:

* + - * 1. Read a CGI string from STDIN, which is in the format of “input\_name\_1=value1&input\_name\_2=value2&input\_name\_3=value3….”. All input\_name\_j and valuej are CGI encoded.
        2. Accept one argument (i.e. arg1) which specifies one of input\_name\_j. Valuej should be the number of Fibonacci numbers to display in browser.

1. getInput(arg1) then retrieve corresponding valuej from the decoded CGI string.

You need to provide modified implementation in labsupport.

Please refer to “item 3”, “item 4” and “sub-item 4 of item 5” of my CGI workshop (<http://learn.ouhk.edu.hk/~t441051/s212/workshop/cgi_workshop/index.html>)

q6a. This question tests your understanding on using C programming to solve problem.

* + - Programming hints.
      * 1. input to your program : name of data file(can be hardcoded inside your program), 3 keys
        2. output : number of occurrence of each key.
        3. Define what data structure you need:
      * There are 3 key inputs, each are numeric character(i.e.1,2,…,49). What data type should be used? Should we group them as an array (in\_key[3]) ?
      * You need to find out occurrence of each key. What data type should be used for number of occurrence? Should we group them as an array (in\_key\_count[3]) ?
      * You need to read the data file line-by-line. So, you need char line[256] to temporarily store the line
      * You may need to counters. Int i, j;
      * You need to open the data file, so a FILE pointer is needed.
        1. You need to read the 3 keys as inputs to the program when it executes. You can choose one of 2 ways: command-line argument or scanf(). For command-line argument, refer to

<https://www.tutorialspoint.com/cprogramming/c_command_line_arguments.htm>

* + - * 1. Each element of in\_key\_count[] must be initialized to zero. What loop can be helpful?
        2. You need to open the data file using fopen() and process

The file is opened to read. The data file is text file. What mode should be used when opening file?

The size of file should not be limited. You should not use for loop to read. What loop should be used if you don’t know the file size in before?

To read a line, you need fgets().

After reading a line as string, you need to read 3 data one-by-one using sscanf(). Each read data should be compared with the 3 keys. If matched, the corresponding in\_key\_count[x] should be incremented by one. Do you think “if” statement is needed? Do you think a 2-level for loops is needed?

* + - * 1. Output to monitor using printf(). Now occurrence of each key is stored in in\_key\_count[]. A for loop may simplify your work.
        2. This URL is helpful <http://learn.ouhk.edu.hk/~t441051/s212/workshop/c_program_beginner_workshop/index.html>
* **Submission**: Don’t include any source code in your \*.doc. Just tell me location of your files, source code and compiled binary, in your labsupport account. Set appropriate permission for compiled binary.
* Supplement:

1. The database file listed in the question is just a sample to help students better understand the question. Your program should not be limited to this sample file.
2. Your program should be:
   * + - * able to handle any size of database file (i.e. number of lines is not fixed.)
         * each line of database file contains exactly 3 numeric data (1 to 49), separated by space.
         * occurence of data in each line is not limited (eg. one line may contain number "2" one, two or three times)
         * able to input 3 numeric keys, separated by space
         * Output number of occurence of each input key in the database file

q6bYou need to develop 2 programs:

* 1. one is program6b.html and
  2. the other is the program6b.cgi.c

For program6b.html, the form should contain 3 input fields to allow users to input 3 keys.

For program6b.cgi.c, you can modify program for Q5, so that it can returns the results to the browser. You can hard-code location and filename of database file inside program6b.cgi.c.

* **How to test:** I will test from INTERNET. So, locate your files (\*.html, \*.cgi) to appropriate directory and set appropriate permission rights
* **Submission**: Don’t include any source code in your \*.doc. Just tell me location of your files, source code and compiled binary, in your labsupport account.

**Q3** This question tests your understanding on using C structure to solve problem.

**Programming specification**:

1. The program reads 3 command line arguments. The 3 arguments are year, month, day, to display in American format.
2. When the program runs without any argument or with incorrect number of arguments, it outputs its running syntax.
3. It validates for meaningful input. If any of argument is NOT a valid number, it displays corresponding invalid messages.

* For example, if year=a, display “invalid year.It must be a number of 4 digit”
* For example, if month=13, display “invalid month”
* For example, if day=40, display “invalid day”
* It also checks for correct month and day for leap year.
* …

1. If inputs are valid, it display corresponding date in American format
2. The program needs to use C structure to store the corresponding input values.

**Programming hints**.

1. For command line argument reading, please refer to P.40 of my “s212\_2017\_Surgery5\_C\_programming.ppt” and demo file **inout\_cmd.c**
2. For use of C structure, please refer to P. 46 of my “s212\_2017\_Surgery5\_supplementary\_C\_programming.ppt”
3. For use of C pointer, please refer to
   1. P. 31 to 35 of my “s212\_2017\_Surgery5\_supplementary\_C\_programming.ppt”
   2. my demo file **pointer.c** in “s212\_2017\_Surgery5\_supplementary\_C\_programming\_demo\_program”
   3. <https://www.tutorialspoint.com/cprogramming/c_pointers.htm>
4. If number of inputs is correct, assign them to your C-structured record.
5. There should be a number of if-else checking for validation. Each if-else corresponds to checking for one type of invalid input. For any invalid input, your program exits and displays corresponding error messages.
6. If all inputs are valid, you can display corresponding date in American format to monitor using printf().

You are required to name your source file and executable file as “**hostname2ip.c**” and “**hostname2ip.exe**” respectively. Place them in the “tma2” directory under your home directory on labsupport.no-ip.org. Set correct permission right to q3.exe so that I can run it when I am marking.

**Q4(a)** You need to identify which kind of client the question is talking about. Then describe the basic steps, described in question, of this type of client.

* The client needs to establish the channel and terminate it. So, what kind of client it should be, conection-oriented or connectless?
* For connection-oriented client, please refer to P.40, 41 of my “s212\_2018\_surgery\_6\_socket\_programming.ppt” file.
* For connectionless client, please refer to P.43, 44 of my “s212\_2018\_surgery\_6\_socket\_programming.ppt” file.

**Q4(b)**

This question includes 3 parts:

1. What is the function of bind()?
2. Why bind() is always required for both stream and datagram server but client side is NOT?
3. Describe which type of client side program requires bind().

For part 1,

* Don’t answer how to bind()
* For its function, please refer to P.45, 47 and 48 of my “s212\_2018\_surgery\_6\_socket\_programming.ppt” file.
* Think following points:
  + - Can you reference to a newly created socket?
    - Can you uniquely identify a newly created socket by IP address and port number?

For part 2, think about:

* Does server always need to handle incoming request?
* Does client always need to handle incoming request?
* Without binding, can client find a way to identify server?
* Which side, client or server, initiates a communication?
* ….

For part 3, please refer to P.43, 44, 47 and 48 of “s212\_2018\_surgery\_6\_socket\_programming.ppt” file.

**Q4(c)** Just refer to **stream\_client.c** in “s212\_2018\_surgery\_6\_demo\_program”. You can easily modify from it after identifying the function of different parts of codes . Don’t use my comment.

* **Submission**: Don’t include any source code in your \*.doc. Just tell me location of your files, source code and compiled binary, in your labsupport account. Set appropriate permission for compiled binary.

You should use following statements:

struct hostent \*srv\_name;

struct sockaddr\_in srv\_info; // server's address information

srv\_name=gethostbyname(argv[1])

srv\_info.sin\_addr = \*((struct in\_addr \*)srv\_name->h\_addr);

printf(" IP= %s \n", inet\_ntoa(srv\_info.sin\_addr));

**---End of hint file ---**