1. 1000 kg of berries are found to be 99% water. After a week, they are 98% water. What is the new mass of berries?
   This is determined by how water being evaporated from berries against the time series. If water evaporation is strictly linear in line with the time series, then the answer is simply (1000*0.98)/0.99 = 989.90. However, in the reality, such an evaporation process could be affected by a number of conditions. As such, we will need to establish a regression model by sampling a set of data. A practical way is to measure the values of mass and the percentage of water against a series of time interval, for example, every hour, and use the collected data to build a predication model. Commonly used approaches are Maximum Likelihood Estimation (MLE), Standard Linear Regression (SLR) using Ordinary Least Squares (OLS) method, and Locally Weight Linear Regression (LWLR). Non-linear regressions might also be used to cope with different kinds of problems. In Q1 folder, I included demonstrative implementations of SLR and LWLR written in Python. Once regression model is established, we can use it for predication. Predicated value $\hat{y}$ is often assigned with a range value δ which is derived from sample standard deviation, and is used to define the range $[\hat{y} - \delta, \hat{y} + \delta]$ that the real value is likely to fall in.

2. How would you monitor a PostgreSQL server to detect SELECTs which are running slowly? Having identified a poorly performing SELECT, how would you analyse it with a view to improving it?
   We normally use EXPLAIN/EXPLAIN ANALYZE command from PostgreSQL shell to monitor the runtime of a SQL query statement over a large scale of data set. If the reported runtime is out of our satisfaction, we will need to reconstruct the SQL Query structure for the purpose of optimisation.

   There are a number of ways we might need to consider. First, when we design a database and its tables, we might also design the uses of index from table fields. Index provides a sorted way for accommodating data which provides a fast way for data query. However, this does not mean that we can abuse the use of index. Since index always performs sorting process, it will bring in additional cost for inserting/deleting data.

   Second, we might consider narrowing the searching scopes by using WITH statement. This is often accomplished by some initial studies

about the data distribution. For example, suppose we have a database that accommodates a large set of persons where 1/3 is between 20 and 29, 1/3 between 30 and 39, and the rest 40+. Among all these people, roughly 50% ladies and 50% gentlemen. They all have different careers. Let say the table is indexed by age. Suppose we want to query persons whose job is software engineer, she is a lady and the age is between 20 and 29. By using "SELECT * FROM table WHERE career='SE' AND sex='F' AND age BETWEEN 20 AND 29;", it is likely that the query will go through the entire table to find the solution. We might consider the use of "WITH persons as (SELECT * from table WHERE age BETWEEN 20 and 29) SELECT * from persons WHERE sex='F' and career='SE';". The WITH statement would reduce the searching space to 1/3 which will largely improve the query performance.

3. A text file contains a list of numbers, one per line. Write a shell pipeline which will print out the five most commonly occurring numbers, one per line without surrounding whitespace, in descending order. Use of perl, awk, or any other programming language should be avoided.

   I extended the work by sorting most frequently occurred words from a file. Given a text file test.txt, it can be performed as:

```
cat test.txt | tr '[:space:]' '[\n*]' |
               tr -d '[:punct:]' | grep -v "^\s*$" |
               sort | uniq -c | sort -bnr
```

   I have put the corresponding files in folder Q3.

4. Do the same in perl (or any programming language of your choice). Use of backticks, system() etc should be avoided.

   Code is written in Python and saved in Q4.

5. If a physically inaccessible computer attached to your local network segment is firewalling all IP traffic, how would you find out if it is up or not? [Hint: this question is very precisely worded.]

   nmap 192.168.1.0/24?

6. A videoconferencing application, running on a mobile device, is consuming 80% CPU, and causing the battery to drain rapidly. What steps might you take to improve battery life? [You have the source code for the application, so you are able to investigate/change both the application itself and its environment]

   I have no experience upon developing mobile applications but would like to discuss my views upon reducing such costs. I believe, in the most case, the system's resources are vastly consumed via 1) data exchange and 2) data presentation. For "data exchange" I mean that mobile app fetches or sends data from/to external devices. This could include GPS navigation, Bluetooth device usage and data stream flow between mobile and the backend server, etc. As such, to reduce such costs, we might perform the following steps. a) switch off GSP and Bluetooth devices; b) always perform data compression before sending it out to reduce the size of data flow; c) instead of fetching data periodically, try to cache data to the backend server as much as we can; d) investigate and determine an optimised data sending/fetching cycle.

   Data presentation determines how a received data to be present to the mobile device. Those that largely consume system's resources are images display, video and audio plays. As such, to reduce the cost, one way I might consider would be reduce the number of frames per second. We need to find a way to determine how we can play video in the speed that would not degrade visual effect too much but can reduce the cost upon consuming system's resources. It might also be considered to preprocess images/videos to make them more coarse-gain without degrading their quality too much.

7. Implement the following function in C:

```
/**
 * Reverse the order of an array of 32-bit integers
 *
 * @v array Array of integers
 * @v count Number of elements in the array
 *
 * This function simply reverses the order of the elements in an
```

```
 * array of 32-bit integers. For example, suppose that we have:
 *
 * uint32_t array[3] = { 0x12345678, 0xdeadbeef, 0xf00df00d };
 *
 * reverse_array ( array, 3 );
 *
 * then array[] should contain { 0xf00df00d, 0xdeadbeef, 0x12345678 }.
 */
void reverse_array ( uint32_t *array, unsigned int count ) {
// Your code goes here


    uint32_t tmp;
    for (size_t i=0; i<count/2; i++) {
        tmp = array[i];
        array[i] = array[count-i-1];
        array[count-i-1] = tmp;
    }
}
```

8. What would the following C code print out?

```
#include <stdio.h>
int main ( int argc, char *argv[] ) {
    int x = 1;
    printf ( "%d %d %d\n", ++x, x, x++ );
    return 0;
}
```

   This is really a bad coding style as it leaves the evaluation order determined by the compiler being used, which, by no means, you will conclude what output you can get. Now from my machine (Ubuntu gcc version 5.4.0), it produces 3 3 1. It looks like the compiler process x++ first (which print x=1 but increase x by 1 afterwards), and then ++x (which increase x by 1 first and then output to print) and last x.

9. Identify as many problems as you can find in the following C code fragment:

```
extern int process_file ( char *data, size_t len );
#define BUFSIZE 4096

int read_file ( int fd ) {
    char buf[BUFSIZE];
    ssize_t read_len;
    size_t len = 0;
    char *data = NULL;
    char *tmp;
    while ( ( read_len = read ( fd, buf, BUFSIZE ) ) > 0 ) {
        tmp = malloc ( len + read_len );
        memcpy ( tmp, data, len );
        memcpy ( tmp + len, buf, read_len );
        free ( data );
        data = tmp;
        len += read_len;
    }
    process_file ( data, len );
    return 0;
}
```

This program will cause memory leakage. I will point out problems as follows:

- Don't use ssize_t until you are sure that the returned value can be negative. However, in this program, we would not expect reading a file will return a negative value;
- tmp should be set to point to NULL to avoid dangling pointer.
- The memcpy(tmp, data, len) function copies 'len' bytes from 'data' to 'tmp' but, in the first loop, 'data' has not been created yet. 'len' is also 0;
- After copy is complete, it calls for free(data) to release memory. Again, in the first loop, data has not been created yet. data is NULL;
- After second loop, 'data' starts to point to dynamically created memory, and, by using free(data) to release memory. But, since in the first loop, malloc(len+read_len) dynamically created a heap

and no operation is performed to release the resource, it causes memory leakage;

10. Write and host a single web-page that should run on a smart phone, implementing (in front-end javascript, not server-side) a basic calculator function. This does not need to be beautiful, but must support at least the following operations: + - * / % ^ SQRT. [By % and ^, I mean "modulo" and "to the power of"] It must detect and notify of illegal input (e.g. sqrt(-1)). Serve it up over https. [Hint: letsencypt is a easy way to do the https part; the javascript should be less than 100 lines, and not rely upon external frameworks or libraries]

    This question looks like to test parsing arithmetic expression using javascript. It can be implemented in the same way as discussed in Q12. In terms of detecting illegal input, it can be easily achieved by using try/catch statement. I am not quite familiar with javascript so would skip the implementation part.

11. Write a script to implement the game "Fizz Buzz". Counting from 1 to 100, print each number in turn, on a new line. BUT...

    - If the number is divisible by 3, or contains a digit 3, print "Fizz" instead of the number;
    - Likewise, for 5s, print "Buzz".
    - If it satisfies both 3 and 5, "Fizz Buzz".

    The code is written in Python and saved in Q11 folder.

12. Convert all letters in the following text, and then evaluate the equation.

    - 'a' to 'z' = 0 to 25 (one to two digits)
    - capital letters = letter number + 1 (again a digit)
    - ', ' = - (coma and space is treated as minus)
    - '. ' = * (dot and space is treated as multiply)
    - ' ' = + (just space is treated as plus)

    [Hint: you are supposed to write a short program to do this... not do it by hand!]

"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum."
(For the purpose of typesetting, the lines have been wrapped. Assume that all whitespace is to be treated as a single space, and that the leadingtrailing spaces/punctuation can be ignored. Work in base 10.)

This question looks like to test the skills upon how to parse a text file into a set of strings. By tokening a text file and transforming these tokens into a predefined format, the work aims to establish an Abstract Binary Searching Tree (ABST). In this work, I have written a program to transform a text file into a list of tokens by considering the rules applied for transformation. These tokens are afterwards concatenated to derive an arithmetic expression.

I did not implement an arithmetic expression parser as there are a lot of open resources that can be used for the parsing purpose. A recommended app can be downloaded from `http://www.partow.net/programming/exprtk/index.html`. The code for tokening text is saved in Q12 folder.

13. Given an array of n points (x,y) in the 2D Cartesian plane, write a short function to find the smallest possible circle that encloses them all. Specify that circle as (x,y,r), i.e. center co-ordinates and radius. [You can treat a point that touches the circle as being "within" it. You should also explain the algorithm you use; a perfect solution to this problem is non-trivial. Since this is a test of programming, rather than mathematics, it is more important that the code you write corresponds to the algorithm you document, than that the algorithm itself is optimal.]

    I don't want to lie as I have come across this question before. This is known as "Smallest-circle problem" `https://en.wikipedia.org/wiki/Smallest-circle_problem`. From 2007 to 2012, I was invited by a Chinese university to give lecture for an MSc course "Algorithm Design and Analysis" where, for this example, I demonstrated students

on how "Welzl's algorithm" can be used to recursively build a solution. The C/C++ code is included in Q13. It was not written by me but by somebody else. Another example I demonstrated was "the closest pair of points" where I showed to students how divided-and-conquer method can be applied to solve the problem with the complexity of $nlog(n)$.