ENGG1340 Computer Programming II / COMP2113 Programming Technologies
Assignment 1
Deadline: 17 March (Friday) 23:59

If you have any questions, please post to the Moodle discussion forum on Assignment 1.

Total marks: 100 marks

General Instructions

Read the instructions in this document carefully.

In this assignment you will solve 3 tasks and a tester would automatically test your submitted program. So if your submitted files and program outputs do not conform to our instructions given here, your programs cannot be evaluated and you will risk losing marks totally.

Sample test cases are provided with each task in this document. Note that the test cases may or may not cover all boundary cases for the problem. It is also part of the assessment whether you are able to design proper test cases to verify the correctness of your program. We will also use additional test cases when marking your assignment submission.

Input and output format

Your C++ programs should read from the **standard input**. Also, your answer should be printed through the **standard output**. If you failed to follow this guide, the tester may not be able to give a score for your program. Additionally, you should strictly follow the sample output format (including space, line breaker, etc.), otherwise, your answer might be considered as wrong.

How to use the sample test cases for problems 2 and 3?

Sample test cases in text file formats are made available for you to check against your work. Here's how you may use the sample test cases. Take Question 3 test case 3 as an example. The sample input and the expected output are given in the files input3_3.txt and output3_3.txt, respectively. Suppose that the object code of your program is named "3", do the followings at the command prompt of the terminal to check if there is any difference between your output and the expected output.

```
./3 < input3_3.txt > myoutput.txt
diff myoutput.txt output3_3.txt
```

Testing against the sample test cases is important to avoid making formatting mistakes. The additional test cases for grading your work will be of the same formats as the sample test cases.

Coding environment

For Problem 2 and Problem 3, you must make sure that your C++ program can compile, execute and generate the required outputs on our standard environment, namely, the gcc C++11 environment we have on the CS Linux servers (academy*). You may not get any mark if your program cannot be executed in the standard environment. Make sure the following compilation command is used to compile your programs:

```
g++ -pedantic-errors -std=c++11 [yourprogram].cpp
```

As a programmer/developer, you should always ensure that your code can work perfectly as expected on a target (e.g., your client's) environment, not only on yours.

While you may develop your work on your own environment, you should always try your program (compile & execute & check results) on our standard environment before submission.

Submission

Name your shell script (for Problem 1) and C++ programs (for Problem 2 and Problem 3) as the following table shows and put them together into one directory. Make sure that the folder contains only these source files (*.sh / *.cpp) and no other files. **Compress this directory as [uid].zip file where [uid] is your university number** and check carefully that the correct files have been submitted. We suggest you to download your submitted file from Moodle, extract them, and check for correctness. **You will risk receiving 0 marks for this assignment if you submit incorrect files.** Resubmission after the deadline is not allowed.

Filename	Description
1.sh	Problem 1
2.cpp	Problem 2
3.cpp	Problem 3

Late submission

If you submit within 3 days after the deadline, there will be 50% mark deduction. After that, no mark will be given.

Evaluation

Your code will be auto-graded for technical correctness. In principle, we use test cases to benchmark your solution, and you may get zero marks for not being able to pass any of the test cases. Normally partial credits will not be given for incomplete solution, as in many cases the logic of the programs are not complete and an objective assessment could be difficult. However, your work may still be considered on a case-by-case basis during the rebuttal stage.

Academic dishonesty

We will be checking your code against other submissions in the class and from the Internet for logical redundancy. Please be reminded that no matter whether it is providing your work to others, assisting others to copy, or copying others will all be considered as committing plagiarism and we will follow the departmental policy to handle such cases. Please refer to the course information notes for details.

Getting help

You are not alone! If you find yourself stuck on something, post your questions to the course forum, send us emails or come to our support sessions. We want this assignment to be rewarding and instructional, not frustrating and demoralizing. But we don't know when or how to help unless you ask.

Discussion forum

Please be careful not to post spoilers. Please don't post any code that is directly related to the assignments. However, you are welcome and encouraged to discuss general ideas on the discussion forums. If you have any questions about this assignment you should post them in the discussion forums.

A. Shell Programming [30%]

A certain university has installed smart card scanners at the gateways. All scanners are connected to a central backend system that stores the access records of staff and students. The system generates a log file every day, each line in the log file represents a transaction made through the gateway and has the following format:

```
[Timestamp],[SmartcardID],[GatewayName],[Enter/Exit]
```

- `Timestamp` has the format `YYYY-MM-DD_HH:MM:SS`.
- `SmartcardID` is a 10-digit value identifying a smartcard user.
- `GatewayName` is the name of a gateway; it may consist of spaces.
- `Enter/Exit` represents whether the user is entering or exiting the university.
- Comma "`,`" is used as field separator in the log file.
- The log files have "`Log_`" as prefix in their filenames, followed by the date in `YYYY-MM-DD`, followed by "`.log`" as file extension.
- You can assume that the transactions are sorted in ascending order of the timestamps.

Write a shell script, *1.sh*, which reads all log files (file name ending in .log) in the current folder and ~~report~~ reports the most popular gateway matching a specific timestamp. Note that multiple gateways can have equal popularity.

The script takes one command line argument, which is a timestamp prefix of records to be extracted.

The script should display the names of gateways entered by the greatest number of distinct smartcard users, preceded by the corresponding distinct smartcard user count, in a descending lexicographical order of gateway names. It should print "No records found" if there is no gateway to output.

You can assume that an argument will always be specified when your script is executed.

Sample input and output
Suppose there are two log files, Log_2023-02-13.log, and Log_2023-02-14.log.

Log_2023-02-13.log:

```
2023-02-13_12:18:44,3035712571,West Gate,Enter
2023-02-13_12:18:44,3035712571,West Gate,Exit
2023-02-13_12:18:46,3034421231,East Gate,Enter
2023-02-13_12:18:48,3033489422,East Gate,Enter
2023-02-13_12:18:59,3035712571,West Gate,Enter
2023-02-13_13:09:29,3032144216,North Gate,Enter
2023-02-13_13:02:44,3036761664,North Gate,Enter
2023-02-13_13:11:32,3035743551,West Gate,Enter
2023-02-13_13:12:10,3034143597,West Gate,Enter
```

Log_2023-02-14.log:

```
2023-02-14_12:00:01,3035743551,West Gate,Enter
2023-02-14_12:20:32,3035573708,East Gate,Enter
2023-02-14_12:35:24,3036954490,North Gate,Enter
2023-02-14_13:11:42,3035743551,West Gate,Exit
2023-02-14_13:13:31,3035743551,East Gate,Enter
```

```
2023-02-14_13:22:01,3033743999,South Gate,Exit
2023-02-14_14:42:11,3036831543,West Gate,Enter
2023-02-14_14:51:42,3035743551,East Gate,Exit
```

Some sample execution of the script and the corresponding output is showed below. Input is highlighted.

```
$ ./1.sh 2023-02-13_12:18:48
1 East Gate
$ ./1.sh 2023-02-13
3 West Gate
$ ./1.sh 2023-02-14
2 West Gate
2 East Gate
$ ./1.sh 2023-02
4 West Gate
4 East Gate
$ ./1.sh 2022
No records found
```

B. C++ Programming [70%]

Problem 2 [30%]:

Write a program *2.cpp* to read two integers M and N, and then print all composite numbers in the range [M, N]. A composite number is a natural number greater than 1 that has more than two factors (other than 1 and itself).

Sample Test Cases:

User inputs are shown in blue.

2_1

```
1 10
4
6
8
9
10
```

3_2

```
30 50
30
32
33
34
35
36
38
39
40
42
44
45
46
48
49
50
```

3_3

```
70 100
70
72
74
75
76
77
78
80
81
82
84
85
86
87
88
90
91
```

92
93
94
95
96
98
99
100

Problem 3 [40%]:

Write a C++ program *3.cpp* which encrypts and decrypts some input characters using a variation of the Caesar Shifting algorithm detailed below.

Input:

- A line of input s k c_1 c_2 c_3 …, where
  - s is either the character e for encryption, or the character d for decryption
  - k is an integer with exactly 6 number digits. The left-most 2 digits form key k_1. The middle 2 digits form key k_2. The right-most 2 digits form key k_3. For example, if k is 226963, k_1, k_2 and k_3 become 22, 69 and 63 respectively. k_1, k_2 and k_3 will be used for encryption or decryption as described below.
  - c_1 c_2 c_3 … is a sequence of space separated characters, ended by !, to be encrypted or decrypted.
  - c_1 will be encrypted with k_1. c_2 will be encrypted with k_2. c_3 will be encrypted with k_3. c_4 will be encrypted with k_1, c_5 will be encrypted with k_2, and so on so forth until the last character is reached.

Output:

- The encrypted/decrypted message ended by !. There is a space between two consecutive characters.

Algorithm:

To encrypt (decrypt) a letter c_i (within the alphabet A-Z or a-z) with a shift of k_j positions:

1. Let x be c_i's position in the alphabet (0 based), e.g., position of B is 1 and position of g is 6.
2. For encryption, calculate $y = x + k_j$ modulo 26;
   For decryption, calculate $y = x – k_j$ modulo 26.
   Here modulo is equivalent to remainder in mathematics. E.g., 29 modulo 26 = 3.
3. Let w be the letter corresponding to position y in the alphabet. If c_i is in uppercase, the encrypted (decrypted) letter is w in lowercase; otherwise, the encrypted (decrypted) letter is w in uppercase.

A character which is not within the alphabet A-Z or a-z will remain unchanged under encryption or decryption.

Example. Given letter B and $k = 3$, we have $x = 1$, $y = 1 + 3$ mod $26 = 4$, and w = E. As B is in uppercase, the encrypted letter is e.

Requirement:

- Implement a function CaesarShift() which takes in a char c and an int k, where c is the character to undergo Caesar Shifting and k is the number of positions (can be negative) to shift, and return the processed char after Caesar Shifting. The returned char is c if c is not within the alphabet range. The prototype of the function is given by: char CaesarShift(char c, int k).
- You can ONLY use the simple data types char, bool, int, double. In other words, you are not allowed the use other data types or data structures such as strings, arrays, vectors, etc.

Sample Test Cases:

User inputs are shown in blue.

3_1

```
e 112233 !
!
```

3_2

```
e 111213 a B c D e !
L n P o Q !
```

3_3

```
d 211322 D e F g H !
i R j L u !
```

3_4

e 134023 H e l l o E N G G 1 3 4 0 / C O M P 2 1 1 3 !
u S I Y C b a u d 1 3 4 0 / z b a m 2 1 1 3 !

3_5
d 202202 n 3 V 3 D 3 N _ M Y N 3 _ S C _ N 3 L E Q Q 3 N _ M Y N 3 !
T 3 t 3 h 3 t _ k e r 3 _ w a _ r 3 r i o w 3 l _ q w t 3 !

If you face any problems, **please feel free to contact us**!

We are very happy to help you!

We wish you enjoy this assignment! ☺