VIETNAM NATIONAL UNIVERSITY - HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

PROGRAMMING FUNDAMENTALS - CO1027

ASSIGNMENT 1

SHERLOCK
A STUDY IN PINK - Part 2

HO CHI MINH CITY, JANUARY 2022

ASSIGNMENT SPECIFICATIONS

Version 1.0

1 Outcomes

After finishing this assignment, the student is revised and can proficiently use:

- Conditional statements
- Loop statements
- Array and 2-dimensional array
- String processing
- Function and function call
- File read/write operations

2 Introduction

The assignment is adapted from Episode 1, Season 1 of the BBC series Sherlock. This movie is also derived from the Sherlock Holmes novel by author Sir Arthur Conan Doyle.

In part 1, Sherlock and Watson are caught up in a string of murders and suicides. The most recent case was of a woman in a pink dress. This case is different from previous cases in that: the victim scratched the floor with her fingernails and left a message. The results of the scene investigation showed that the victim lost her luggage. With Sherlock's talent, he found the lost luggage. Thereby, he took it to room 221B Baker Street and looked for further traces of the criminal. Watson also returns at the same time and joins Sherlock.

3 Tasks

Students are asked to implement a program in C++ to simulate the solving process of the first case of Sherlock and Watson: A study in Pink, through the tasks described below.

3.1 Task 1: The luggage password (5 points)

The found luggage has a 10-digit password, Sherlock needs to deduce this password before considering follow-up clues. Sherlock searched the outer drawers of the luggage and discovered

3 notebooks. According to him, the victim often takes off her ring before working, she must be a careful person; then why are these notebooks placed outside and not inside? These notebooks are probably left out intentionally by the victim. They contain information about the luggage password in case the victim forgets the password and can find the password again. Thereby, Sherlock reads through the notebooks and begins his investigation. However, Sherlock does not like tedious tasks, he will explain how to decode the password and ask Watson to do the rest. Watson then... asked you, the students, to help him find the password.

3.1.1 The First Notebook

Sherlock looks through the first notebook which pages marked with different numbers from 0 to 9. Only the first line contains a strange string of characters **Freq/First/¡n1¿**. Sherlock guessed that he needed to take the first **n1** numbers and list out their frequencies (Freq=Frequency).

Students are asked to write a function to find the password from the first notebook. The description of the function follows:

- Function name: **notebook1**.
- Input parameters:
  - **ntb1**: a string containing filename, this file contains information of the first notebook.
- Structure of the file **ntb1**:
  - Line 1: a formatted string **Freq/First/¡n1¿**, in which **¡n1¿** has 3 numeric characters. **¡n1¿** will represent an integer in the range $[1, 999]$. For example, $n1 =$ "001" represents the integer 1; $n1 =$ "010" represents the integer 10; $n1 =$ "123" represents the integer 123.
  - Line 2: contains multiple digits, the digits are separated by a space. This number of digits is guaranteed to be greater than or equal to **¡n1¿** if **¡n1¿** is a valid string.
- Function requirements: The function reads data from the file **ntb1**. If **¡n1¿** is an invalid string, the function does nothing and returns the value "0000000000". If the string **¡n1¿** is valid, we count the number of appearances for each numeric character ('0' to '9') in the first **¡n1¿** digits of Line 2. Each number will be listed consecutively in increasing order of the numeric characters from '0' to '9'. If a digit does not appear in the first **¡n1¿** digits of Line 2, we consider the number of appearances to be 0. After counting, if the number of appearances of a numeric character is a number greater than 10, we keep only the

last digit of the number of appearances. The function returns a string of 10 characters, respectively the number of appearances for each numeric character.

Example 1: Examples of **¡n1¿** being an invalid string:

- **Freq/First/11x**: invalid because the last character is 'x', not a digit.
- **Freq/First/000**:invalid because when parsed to integer will be zero, not in the range $[1, 999]$.

The function returns the value "0000000000" in those two cases.

Example 2: The file **ntb1** has the following content:

1	Freq/First/014
2	1 1 1 1 1 1 1 1 1 1 9 9 3 5 5 5

The frequence is:

0	1	2	3	4	5	6	7	8	9
0	$11 \rightarrow 1$	0	1	0	0	0	0	0	2

The function returns value: "0101000002".

3.1.2 The Second Notebook

After solving notebook 1, Sherlock looks to notebook 2. The first line of the notebook is a positive integer. The following lines are strings of varying lengths; however, Sherlock sees many of the words "pink" appear in these strings. Sherlock guesses he needs to find the number of occurrences of the string "pink" in this notebook.

Students are asked to write a function to find the password from notebook 2. The description of the function is as follows:

- Function name: **notebook2**.
- Input parameters:
  - **ntb2**: a string containing filename, this file contains information of the second notebook.
- Structure of the file **ntb2**:

  - Line 1: a string **¡n2¿** represents an integer in the range $[5, 100]$, this string has 5 numeric characters.
  - **¡n2¿** next lines: each line is a string of random characters (possibly with a space character).

- Function requirements: Function reads data from the file **ntb2**. If **¡n2¿** is an invalid string, the function does nothing and returns the value "1111111111". If the string **¡n2¿** is valid, we count the number of times the string "Pink" or "pink" appears in the next **¡n2¿** line and call this number by cntP. We do the following transformation:
  - Step 1: If cntP has less than 5 digits, then update:

$$\text{cntP} = \text{cntP}^2$$

    Otherwise, do nothing.
  - Step 2: If cntP does not have exactly 10 digits, we add to the last of cntP some characters "9" until it have 10 digits.

  The function returns the value cntP with exactly 10 digits.
  Note: The testcases will make sure the number of occurrences is a number not exceeding 9 digits.

Example 3: Examples of **¡n2¿** as invalid strings:

- **12a**: invalid because this string is only 3 characters, the last character is 'a', not a digit.
- **00003**: this string has 5 digits, but the corresponding integer representation is 3, not in the range $[5, 100]$.

Example 4: Given the file **ntb2** with the following content:

Line	Contents	Number of "pink" or "Pink"
1	00005	
2	pink–Pink	2
3	Hello Pink 99Pinky77	2
4	I'm doing an easy Assignment, my Pink	1
5	Really?! 1Pink2 3pink4 5Pink6 7pink8 9pink9	5
6	Yep./-good luck and good night, pink	1

According to the above table, we have cntP = 11. Then we transform the result:

- Step 1: Since cntP has only 2 digits, cntP is updated to

$$\text{cntP} = 11^2 = 121$$

- Step 2: Since cntP has only 3 digits, we add 9's to the end:

$$\text{cntP} = 11^2 = 121999999$$

The function returns the value: "121999999".

3.1.3 The Third Notebook

Notebook 3 contains the information of a 10x10 square matrix consisting of only integers. The back of the notebook says, "Prime above, Fibo below". Students are asked to write a function to find the password from notebook 3 with the following description:

- Function name: **notebook3**.
- Input parameters:
  - **ntb3**: a string containing filename, this file contains information of the third notebook.
- Structure of the file **ntb3**:
  - File contains 10 lines.
  - Each line contains 10 integers (possibly negative), integers are separated by a character '—'.
- Function requirements: The function reads data from the file **ntb3** and stores it in a 2D array. If an integer of 2D array is negative, then we write its opposite in the array. We perform the following transformations:
  - Step 1: increment all elements above the main diagonal to the value of the nearest prime. If a number is already prime, then keep it. The main diagonal of a matrix is the diagonal starting from the element in the upper left corner to the element in the lower right corner.
  - Step 2: increment all elements below the main diagonal by a Fibonacci number. A Fibonacci number is a number in the Fibonacci sequence. This is an infinite sequence of natural numbers starting with 1 and 1, then the following numbers will be the sum of the 2 preceding numbers.

- Step 3: For each row of the 2-D array, sort the last 3 integers of the row into the **not decreasing** ordered integers.

- Return value: Given i_0 to i_9 which is the position (from 0) of the largest element of row 0 to row 9 respectively. If there are many elements equal to the largest value, take the position of the last element. The function returns a concatenated string of numbers from i_0 to i_9.

Example 5: Assume the file ntb3 has the following content:

Line	Contents
1	1—11—1—12—1—1—1—11—12—1
2	1—11—1—12—1—1—1—12—1—11
3	1—11—1—12—1—1—1—1—1—1
4	1—11—1—12—1—1—1—1—1—1
5	1—11—1—12—1—1—1—1—1—1
6	1—11—1—12—1—1—1—1—1—1
7	1—11—1—12—1—1—1—1—1—1
8	1—11—1—12—1—1—1—1—1—1
9	1—11—1—12—1—1—1—1—1—1
10	1—11—1—12—1—1—1—1—1—1

Result after performing steps 1 and 2:

Line	Contents
1	1—11—2—13—2—2—2—11—13—2
2	1—11—2—13—2—2—2—13—2—11
3	1—13—1—13—2—2—2—2—2—2
4	1—13—1—12—2—2—2—2—2—2
5	1—13—1—13—1—2—2—2—2—2
6	1—13—1—13—1—1—2—2—2—2
7	1—13—1—13—1—1—1—2—2—2
8	1—13—1—13—1—1—1—1—2—2
9	1—13—1—13—1—1—1—1—1—2
10	1—13—1—13—1—1—1—1—1—1

The result after doing step 3 is:

Line	Contents
1	1—11—2—13—2—2—2—2—11—13
2	1—11—2—13—2—2—2—2—11—13
3	1—13—1—13—2—2—2—2—2—2
4	1—13—1—12—2—2—2—2—2—2
5	1—13—1—13—1—2—2—2—2—2
6	1—13—1—13—1—1—2—2—2—2
7	1—13—1—13—1—1—1—2—2—2
8	1—13—1—13—1—1—1—1—2—2
9	1—13—1—13—1—1—1—1—1—2
10	1—13—1—13—1—1—1—1—1—1

The function returns the value: "9931333333".

3.1.4 Combination of passwords

After having 3 passwords from 3 notebooks, Sherlock still does not know which will be the correct password, or maybe it requires a combination of passwords. Given pwd1, pwd2, pwd3 respectively the password obtained when decrypting 3 notebooks **ntb1, ntb2, ntb3**. Students implement the function with the following descriptions:

- Function name: **generateListPasswords**.
- Input parameters:
  - `pwd1`: A string containing the password decoded from notebook 1.
  - `pwd2`: A string containing the password decoded from notebook 2.
  - `pwd3`: A string containing the password decoded from notebook 3.
- Function requirements: Call `g(p1, p2)` is a function that performs a combination of two passwords p1 and p2 by: performing the addition of each digit at each corresponding position from left to right. If the added result of previous position has a remainder, the remainder will be added to the next position. If the last addition has a remainder, we ignore this remainder.
- Return values: Returns a string representing the possible passwords when a combination of 3 passwords is found, each password separated by a comma. The following are the passwords and combinations that need to be created, the `g` function is put in the "¡¿" pair with the meaning that it is necessary to find the result of the function `g` before

placing it in the return string:

1. pwd1
2. pwd2
3. pwd3
4. <g(pwd1,pwd2)>
5. <g(pwd1,pwd3)>
6. <g(pwd2,pwd3)>
7. <g(<g(pwd1,pwd2)>,pwd3)>
8. <g(pwd1,<g(pwd2,pwd3)>)>
9. <g(<g(pwd1,pwd2)>,<g(pwd1,pwd3)>)>

We will join the above 9 passwords together, between the passwords there is 1 character ',' (comma). The string after concatenation is the return value of the function.

After trying the above passwords, Sherlock can finally open the luggage. He searched through the items in the luggage. There is a small laptop, but unfortunately, it is also password protected. After a while, Watson wonders what is the thing that Sherlock looking for. Sherlock explains the victim will carry his cell phone. The phone was not at the scene, and neither was it in the luggage. So, it's most likely with the criminal. Sherlock tells Watson to send a text message to the victim's phone, and tells him that she just woke up from fainting and didn't know what happened. After that, they made an appointment with the person holding the phone to meet at an address.

3.2 Chase the taxi

After making an appointment to meet the person holding the phone, Sherlock is confident that he would be worried to hear that the victim was still alive if it were a criminal. The criminal will come to the arrangement point to see the actual condition of the victim. Sherlock and Watson went to a roadside shop about 5m away from the meeting point and watched together. A taxi came and stopped there, the person sitting in the taxi looked out searching. When this person accidentally looks in Sherlock's direction, the car starts up and leaves. Sherlock knows the streets of the city he lives in well. He and Watson ran through the shortcuts and chased the taxi.

Students are asked to write the following function to describe this process. The function information is as follows:

- Function name: **chaseTaxi**.
- Input parameters:
  - `arr`: Contains the address of the first element of a fixed size 100x100 2-dimensional array. Each element of the array is an integer. This 2D array represents the map on which the Taxi runs.
  - `points`: The sequence represents the points where Sherlock expected to meet the taxi and intercept it. These points are separated by a character ';' (semi-colon). Each point is represented as:

    $$(p_i, p_j)$$

    where p_i, p_j is the index of that point by row and by column, respectively.
  - `moves`: The sequence contains the moves of the Taxi. Each character is one of the following four letters:
    - 'U': go up (up)
    - 'D': go down (down)
    - 'L': go left (left)
    - 'R': right (right)
  - `outTimes`: String variable passed by reference, used to return the time it took Sherlock and Watson to reach the points.
  - `outCatchUps`: String variable passed by reference, used to return the result of whether Sherlock and Watson have caught up with taxi at points.
- Function requirements:
  1. Via the variable `arr` (same known size 100x100), initialize all elements of the 2-D array to the value -9. This value represents all squares that have not been traversed by Taxi.
  2. Initially, Taxi is at position (0,0). Based on the variable `moves`, student represents the move of Taxi in turn through the cells on the 2-dimensional array corresponding to each character that has been explained. Assume the travel time between 2 adjacent cells is 14 (time unit). When Taxi is at position (0,0), we assign the value to this cell as 0. For each step when Taxi goes to a new cell on the map, we assign the value to that cell equal to the value of the previous cell adding 14. While moving, if Taxi has reached a row/column that is the boundary of the map, and the next step of `moves` is to go outside the map, Taxi will skip the next step and stand still.
  3. For each point in the string `points`, we calculate the time it takes Sherlock and Watson to reach that point. The travel time to a point p_i is equal to the sum of the

time it takes to get to the point $p_i - 1$ and the time it takes to get from the point $p_i - 1$ to the point p_i. However, the travel time to the first point will be calculated as the travel time from position (0,0) to that point. Because Sherlock and Watson are jogging, their travel speed will be slower than Taxi's, this speed is 9 (time unit) when passing 1 cell. Let d be the Manhattan distance between 2 points, the travel time between the 2 points is: $9 * d$. The Manhattan distance between 2 points (x1,y1) and (x2,y2) is calculated as:

$$|x1 - x2| + |y1 - y2|$$

- Return results:
  - The function returns `true` if Sherlock and Watson catch up with Taxi. Otherwise, returns `false`.
  - `outTimes`: The string represents the time it takes Sherlock and Watson to move to the points. The times are joined together in the order of the points and separated by a ';' (semi-colon). If at a certain point, two people have caught up with Taxi, then the next points, we do not need to calculate the travel time to these points. We will replace it with the '-' character.
  - `outCatchUps`: The string represents at each point whether Sherlock catches up to Taxi or not. If so, we represent it with the letter 'Y'. Otherwise, we represent it with the letter 'N'. If this is a point after the point that has caught up with the Taxi, we represent it with the '-' character. These characters are then concatenated into a string, each character separated by a ';' (semi-colon). A point where Sherlock catches a Taxi if the time it takes Sherlock to reach that point does not exceed the time it takes for the Taxi to reach that point.

Example 6: Here is an example of a value of `points`:

- `"(20,30)-(70,90)"`: there are 2 points Sherlock intends to catch up with Taxi. Point 1: row index is 20, column index is 30. Point 2: row index is 70, column index is 90.

Example 7: Here is an example of a value of `moves`:

- `"RRRUDL"`
- The position in turn changes to:
  $(0, 0) \xrightarrow{R} (1, 0)$

$\xrightarrow{R}$ (2, 0)
$\xrightarrow{R}$ (3, 0)
$\xrightarrow{U}$ (3, 0) (at the top edge of the map so can't move up)
$\xrightarrow{D}$ (3, 1)
$\xrightarrow{L}$ (2, 1)

3.3 Laptop Password

After chasing the Taxi, Sherlock and Watson return to the apartment to rest. Sherlock is noticed by the Laptop found in the luggage. Outside the luggage was a card with the victim's email address, he tried using this email for the username, and the password, he tried all the usual passwords but it didn't work. Inspector Lestrade called to remind Sherlock that he had investigated the message left by the victim. This message is the name of the victim's daughter who died a few years ago.

Students are asked to write the following function to describe this process. The function information is as follows:

- Function name: **enterLaptop**.
- Input parameters:
  - `tag`: String containing the name of the file, which contains the information of the tag on the outside of the baggage.
  - `message`: String containing the name of the victim's daughter.
- File's structure **tag**:

Line	Contents
1	Email: ¡victim's email¿
2	Address: ¡n3¿ THT Street

  Where `<n3>` is the string representing an integer in the range $[1, 20]$.
- Funtion requirements: Find the password of the laptop knowing that this password was created by repeating the victim's daughter's name `<n3>` times (see more examples below).
- Return results: Call the found laptop's password **pwdL**. The function will return a string by concatenating the victim's email and **pwdL**, separating these two elements by a character ';' (semicolon): **¡victim's email¿;¡pwdL¿**.

Example 8: Here is an example of the contents of the file **tag**:

Line	Contents
1	Email: pinky@cse.com
2	Address: 3 THT Street

Example 9: Given the daughter's name is **Rachel** and `<n3>` = 3, the laptop's password is **RachelRachelRachel**.
Assume the contents of the file **tag** are the same as the previous example. The function returns the value: **"pinky@cse.com;RachelRachelRachel"**.

3.4 Ending

After opening the laptop, the victim's desktop only has some basic office software and a navigation software. This software has been installed to connect to the victim's phone. Fortunately, the username and password of the software are the same as this information when logging into the laptop. The software started searching, the area on the screen began to shrink and display the address 221B Baker Street. Mrs. Hudson ran up to report a Taxi parked downstairs, the driver asked her to deliver: **"Special taxi for Sherlock Holmes."**. Sherlock suddenly understood everything, he told Watson to stay and continue searching, he needed to go out and take a breather.

What will Sherlock do to deal with the taxi driver at the door, who is most likely the culprit of 4 suicide cases? Will Watson keep up with Sherlock and assist him in defeating this criminal? We'll continue to work on new quests with Sherlock and Watson in part 3 of this assignment.

Have fun doing the assignment!!!

4 Submission

Students submit a file: **studyInPink2.h** in the site "Ky thuat lap trinh (CO1027)_HK212_ALL".

Deadlines for submission are announced at the submission site above. By the deadline for submission, the link will be locked automatically, so students will not be able to submit them late. To avoid possible risks at the time of submission, students MUST submit their papers at least **one hour** before the deadline.

5 Handling fraud

Assignment must be done BY YOURSELF. Students will be considered fraudulent if:

- There is an unusual similarity between the source code of the submissions. In this case, ALL submissions are considered fraudulent. Therefore, students must protect the source code of their assignments.
- Students do not understand the source code written by themselves, except for the parts of the code provided in the initialization program. Students can consult from any source, but make sure they understand the meaning of all the lines they write. In the case of not understanding the source code of the place they refer, students are especially warned NOT to use this source code; instead use what has been learned to write programs.
- Mistakenly submit another student's assignment on your personal account.

In the case of cheating, students will get a 0 for the entire subject (not just the assignment).

DO NOT ACCEPT ANY INTERPRETATION AND NO EXCEPTION!

After each major assignment has been submitted, a number of students will be called for random interviews to prove that the assignment has been done by themselves.

6 Change from previous version

References

[1] A Study in Pink, Wikipedia, `https://en.wikipedia.org/wiki/A_Study_in_Pink`

[2] Sherlock, Season 1 - Episode 1: A Study in Pink, Netflix, `https://www.netflix.com/watch/70174779?trackId=13752289`.

——————————END——————————