ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

Kỹ thuật lập trình - CO1027

Bài tập lớn 1

SHERLOCK
A STUDY IN PINK - Phần 2

TP. HỒ CHÍ MINH, THÁNG 01/2022

ĐẶC TẢ BÀI TẬP LỚN

Phiên bản 1.0

1 Chuẩn đầu ra

Sau khi hoàn thành bài tập lớn này, sinh viên ôn lại và sử dụng thành thục:

- Các cấu trúc rẽ nhánh
- Các cấu trúc lặp
- Mảng 1 chiều và mảng 2 chiều
- Xử lý chuỗi ký tự
- Hàm và lời gọi hàm
- Các thao tác đọc/ghi tập tin

2 Dẫn nhập

Bài tập lớn (BTL) này được phóng tác dựa trên tập 1 mùa 1 của bộ phim Sherlock của đài BBC. Bộ phim này cũng được thực hiện dựa trên cuốn tiểu thuyết Sherlock Holmes của tác giả Sir Arthur Conan Doyle.

Trong phần 1, Sherlock và Watson bị cuốn vào một chuỗi các vụ án mạng tự tử. Vụ án gần nhất là của một người phụ nữ trong trang phục màu hồng. Vụ án này khác với các vụ án trước ở chỗ: nạn nhân dùng móng tay cào lên mặt sàn và để lại một thông điệp. Kết quả điều tra hiện trường cho thấy nạn nhân bị mất hành lý, với tài năng của mình, Sherlock đã tìm thấy hành lý bị mất. Anh đem về căn phòng số 221B đường Baker và tìm kiếm các dấu vết tiếp theo của tên tội phạm. Watson cũng trở về cùng lúc và tham gia với Sherlock.

3 Nhiệm vụ

Sinh viên được yêu cầu xây dựng một chương trình giả tưởng trên ngôn ngữ C++ để mô phỏng lại quá trình giải quyết vụ án đầu tiên của Sherlock và Watson: A study in Pink, thông qua các nhiệm vụ được mô tả bên dưới. Mỗi nhiệm vụ được yêu cầu viết hàm tương ứng, các tham số của cho hàm này sẽ được cho trong mô tả của yêu cầu nhiệm vụ.

3.1 Nhiệm vụ 1: Mật mã của hành lý (5 điểm)

Hành lý tìm thấy được đặt mật khẩu với 10 chữ số, Sherlock cần suy luận ra mật khẩu này mới có thể xem xét các manh mối tiếp theo. Sherlock tìm các ngăn kéo bên ngoài của hành lý và phát hiện 3 cuốn sổ ghi chép. Theo anh, nạn nhân thường xuyên tháo nhẫn trước khi làm việc, cô ấy hẳn là một người cẩn thận, tại sao các cuốn sổ này lại được đặt bên ngoài mà không phải bên trong? Các cuốn sổ này có lẽ là nạn nhân cố tình để ở ngoài, chúng chứa thông tin của mật mã hành lý phòng trường hợp nạn nhân quên mật khẩu thì có thể tìm lại được. Từ đó, Sherlock đọc qua các cuốn sổ và bắt đầu việc điều tra. Tuy nhiên, Sherlock không thích các công việc nhàm chán, anh sẽ giải thích cách giải mã mật khẩu và nhờ Watson làm phần việc còn lại. Watson thì... nhờ các bạn SV cùng giúp anh ấy tìm mật khẩu.

3.1.1 Cuốn sổ 1

Sherlock nhìn qua cuốn sổ đầu tiên, các trang của nó ghi các chữ số khác nhau từ 0 đến 9. Chỉ có dòng đầu tiên ghi một chuỗi ký tự lạ **Freq/First/<n1>**. Sherlock đoán rằng, anh cần phải lấy **n1** số đầu tiên và liệt kê tần số (freq=frequency) của các số này.

Sinh viên được yêu cầu viết một hàm để tìm ra mật khẩu từ cuốn sổ 1, mô tả về hàm như sau:

- Tên hàm: **notebook1**.
- Tham số đầu vào:
  - **ntb1**: Chuỗi chứa tên tập tin, tập tin này chứa thông tin của cuốn sổ 1.
- Cấu trúc của tập tin **ntb1**:
  - Dòng 1: 1 chuỗi có dạng **Freq/First/<n1>**, trong đó **<n1>** có 3 ký tự đều là 3 chữ số. **<n1>** sẽ biểu diễn một số nguyên trong khoảng $[1, 999]$. Ví dụ: $n1 =$ "001" biểu diễn cho số nguyên 1; $n1 =$ "010" biểu diễn cho số nguyên 10; $n1 =$ "123" biểu diễn cho số nguyên 123.
  - Dòng 2: chứa nhiều chữ số, các chữ số cách nhau bởi 1 khoảng trắng. Số lượng chữ số này luôn đảm bảo lớn hơn hoặc bằng **<n1>** nếu **<n1>** là một chuỗi hợp lệ.
- Yêu cầu hàm: Hàm đọc dữ liệu từ tập tin **ntb1**, nếu **<n1>** là một chuỗi không hợp lệ, hàm không làm gì cả và trả về giá trị "0000000000". Nếu chuỗi **<n1>** là hợp lệ, ta liệt kê số lượng các chữ số thuộc cùng 1 loại dựa trên **<n1>** chữ số đầu tiên của Dòng 2. các số lượng này được sắp xếp theo thứ tự tăng dần của mỗi chữ số từ '0' đến '9'. Nếu chữ số nào không xuất hiện trong **<n1>** chữ số đầu tiên của Dòng 2, ta coi số lượng chữ số

đó là 0. Sau khi liệt kê, nếu số lượng này là một số lớn hơn 10, ta chỉ giữ lại chữ số cuối cùng (chữ số ở hàng đơn vị). Hàm trả về chuỗi gồm 10 ký tự, lần lượt là số lượng liệt kê tìm được.

Ví dụ 1: Các ví dụ về <**n1**> là chuỗi không hợp lệ:

- **Freq/First/11x**: không hợp lệ do có ký tự cuối cùng là 'x', không phải là một chữ số.
- **Freq/First/000**: không hợp lệ do khi chuyển qua số nguyên sẽ là số 0, không thuộc khoảng $[1, 999]$.

Hàm trả về giá trị "0000000000" trong 2 trường hợp này.

Ví dụ 2: Tập tin **ntb1** có nội dung như sau:

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

Kết quả liệt kê là:

0	1	2	3	4	5	6	7	8	9
0	11 → 1	0	1	0	0	0	0	0	2

Hàm trả về giá trị: "0101000002".

3.1.2 Cuốn sổ 2

Sau khi giải mã xong cuốn sổ 1, Sherlock xem tới cuốn sổ 2. Dòng đầu tiên của cuốn sổ là một số nguyên dương. Các dòng sau là các chuỗi với độ dài khác nhau, tuy nhiên, Sherlock nhìn thấy trong các chuỗi này có nhiều từ "pink" xuất hiện. Sherlock đoán anh cần tìm số lần xuất hiện của chuỗi "pink" trong cuốn sổ này.

Sinh viên được yêu cầu viết một hàm để tìm ra mật khẩu từ cuốn sổ 2, mô tả về hàm như sau:

- Tên hàm: **notebook2**.
- Tham số đầu vào:
  - **ntb2**: Chuỗi chứa tên tập tin, tập tin này chứa thông tin của cuốn sổ 2.

- Cấu trúc của tập tin **ntb2**:
  - Dòng 1: 1 chuỗi biểu diễn cho một số nguyên dương <**n2**> nằm trong khoảng $[5, 100]$, chuỗi này phải có đúng 5 ký tự, mỗi ký tự này đều phải là một chữ số.
  - <**n2**> dòng tiếp theo: mỗi dòng là một chuỗi các ký tự bất kì (có thể có ký tự khoảng trắng).
- Yêu cầu hàm: Hàm đọc dữ liệu từ tập tin **ntb2**, nếu <n2> là một chuỗi không hợp lệ, hàm không làm gì cả và trả về giá trị "1111111111". Nếu chuỗi <**n2**> là hợp lệ, ta đếm lần chuỗi "Pink" hoặc "pink" có xuất hiện trong <**n2**> dòng tiếp theo. Gọi số lần tìm thấy này là cntP. Ta biến đổi kết quả này qua 2 bước sau:
  - Bước 1: Nếu cntP có số chữ số là ít hơn 5 thì cập nhật:

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

    Ngược lại, không làm gì cả.
  - Bước 2: Nếu cntP chưa có đủ 10 chữ số, ta thêm các chữ số 9 vào cuối cùng cho đến khi đủ 10 chữ số.

  Hàm trả về giá trị là cntP có đúng 10 chữ số.
  Lưu ý: Các testcases sẽ đảm bảo số lần xuất hiện là một số không vượt quá 9 chữ số.

Ví dụ 3: Các ví dụ về <**n2**> là chuỗi không hợp lệ:

- **12a**: không hợp lệ do chuỗi này chỉ có 3 ký tự, ký tự cuối cùng là 'a' không phải là một chữ số.
- **00003**: chuỗi này có 5 chữ số, nhưng số nguyên biểu diễn tương ứng là 3, không nằm trong khoảng $[5, 100]$.

Ví dụ 4: Với tập tin **ntb2** có nội dung như sau:

Dòng	Nội dung	Số chuỗi "pink" hoặc "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

Theo bảng trên, ta có cntP = 11. Sau đó ta biến đổi kết quả:

- Bước 1: Do cntP chỉ có 2 chữ số nên cntP được cập nhật thành

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

- Bước 2: Do cntP chỉ có 3 chữ số nên ta thêm các chữ số 9 vào cuối:

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

Hàm trả về giá trị: "121999999".

3.1.3 Cuốn sổ 3

Cuốn sổ 3 chứa thông tin của một ma trận vuông 10x10 chỉ gồm các số nguyên. Mặt sau của cuốn sổ có ghi "Prime above, Fibo below". Sinh viên được yêu cầu viết một hàm để tìm ra mật khẩu từ cuốn sổ 3 với mô tả như sau:

- Tên hàm: **notebook3**.
- Tham số đầu vào:
  - **ntb3**: Chuỗi chứa tên tập tin, tập tin này chứa thông tin của cuốn sổ 3.
- Cấu trúc của tập tin **ntb3**:
  - Tập tin gồm 10 dòng.
  - Mỗi dòng gồm 10 số nguyên (có thể có số nguyên âm), các số nguyên được ngăn cách bởi 1 kí tự '|'.
- Yêu cầu hàm: Hàm đọc dữ liệu từ tập tin **ntb3** và lưu trữ vào một mảng 2 chiều. Nếu số nguyên đọc được là số âm thì ta ghi số đối của nó vào mảng. Ta thực hiện các bước biến đổi sau:
  - Bước 1: tăng tất cả các phần tử ở phía trên đường chéo chính lên giá trị là số nguyên tố gần nó nhất, nếu số đó đã là số nguyên tố thì giữ nguyên. Đường chéo chính của ma trận là đường chéo bắt đầu từ phần tử nằm ở góc trên trái, đến phần tử ở góc dưới phải.
  - Bước 2: tăng tất cả các phần tử ở phía dưới đường chéo chính lên giá trị là một số Fibonacci. Số Fibonacci là một số nằm trong dãy Fibonacci. Đây là dãy vô hạn các

số tự nhiên bắt đầu bằng 1 và 1, sau đó các số tiếp theo sẽ bằng tổng của 2 số liền trước nó.

- Bước 3: Với mỗi hàng của mảng 2 chiều, sắp xếp 3 số nguyên cuối cùng của hàng thành 3 số nguyên có thứ tự **không giảm**.

- Kết quả trả về: Gọi i_0 đến i_9 lần lượt là vị trí (tính từ 0) của phần tử lớn nhất của hàng 0 đến hàng 9. Nếu có nhiều phần tử bằng với giá trị lớn nhất thì lấy vị trí của phần tử cuối cùng. Hàm trả về chuỗi được ghép bởi các số từ i_0 đến i_9.

Ví dụ 5: Giả sử tập tin **ntb3** có nội dung như sau:

Dòng	Nội dung
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

Kết quả sau khi thực hiện bước 1 và 2:

Dòng	Nội dung
1	1|1|1|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

Kết quả sau khi thực hiện bước 3 là:

Dòng	Nội dung
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

Hàm trả về giá trị: "9931333333".

3.1.4 Tổng hợp các mật khẩu

Sau khi có 3 mật khẩu từ 3 cuốn sổ, Sherlock vẫn chưa biết mật khẩu nào sẽ là mật khẩu đúng, hoặc có thể cần có sự kết hợp giữa các mật khẩu với nhau. Gọi pwd1, pwd2, pwd3 lần lượt là mật khẩu có được khi giải mã 3 cuốn sổ **ntb1, ntb2, ntb3**. Sinh viên hiện thực hàm với các mô tả sau:

- Tên hàm: **generateListPasswords**.
- Tham số đầu vào:
  - `pwd1`: Chuỗi chứa mật khẩu được giải mã từ cuốn sổ 1.
  - `pwd2`: Chuỗi chứa mật khẩu được giải mã từ cuốn sổ 2.
  - `pwd3`: Chuỗi chứa mật khẩu được giải mã từ cuốn sổ 3.
- Yêu cầu hàm: Gọi `g(p1, p2)` là hàm thực hiện tổ hợp 2 mật khẩu p1 và p2 bằng cách: thực hiện phép cộng từng chữ số ở từng vị trí tương ứng theo chiều từ trái sang phải. Phần dư nếu có của kết quả cộng ở vị trí trước sẽ được cộng vào vị trí sau. Nếu cộng ở vị trí cuối cùng có dư thì ta bỏ qua phần dư này.
- Kết quả trả về: Trả về một chuỗi biểu diễn các mật khẩu có thể có khi tổ hợp 3 mật khẩu tìm thấy, mỗi mật khẩu sẽ được phân cách bởi 1 dấu phẩy. Sau đây là các mật khẩu và tổ hợp cần tạo, hàm `g` được bỏ trong cặp dấu "<>" với ý nghĩa cần tìm ra kết quả của hàm `g` trước khi đặt vào chuỗi trả về:

  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)>)>

Ta sẽ nối 9 mật khẩu trên với nhau, giữa các mật khẩu có 1 kí tự ',' (dấu phẩy). Chuỗi sau khi nối là giá trị trả về của hàm.

Sau khi thử các mật khẩu trên, cuối cùng, Sherlock cũng mở được hành lý. Anh lục soát các đồ vật trong hành lý. Đáng chú ý trong số này là một chiếc laptop nhỏ, nhưng không may là nó cũng bị đặt mật khẩu. Một lúc sau, Watson thắc mắc về việc Sherlock vẫn đang tìm kiếm gì đó. Sherlock giải thích, nạn nhân sẽ cầm theo điện thoại của mình. Chiếc điện thoại không có ở hiện trường, nhưng cũng không có trong hành lý. Vậy, nó rất có thể ở chỗ tên tội phạm. Sherlock bảo Watson hãy gửi một tin nhắn đến điện thoại của nạn nhân, Watson hãy nhắn rằng mình vừa tỉnh dậy sau khi bị ngất và không biết có chuyện gì xảy ra. Sau đó, cả hai hẹn người đang giữ điện thoại đến gặp tại một địa chỉ để lấy lại đồ.

3.2 Nhiệm vụ 2: Đuổi theo taxi (3 điểm)

Sau khi nhắn tin hẹn gặp người đang giữ điện thoại, Sherlock chắc chắn rằng, nếu đó là tên tội phạm, hắn sẽ lo lắng khi nghe nạn nhân vẫn còn sống. Tên tội phạm sẽ đến điểm hẹn để xem tình trạng thực tế của nạn nhân. Sherlock và Watson đến một cửa hàng bên đường cách điểm hẹn khoảng 5m và cùng theo dõi. Một chiếc taxi đến dừng tại đó, người ngồi trên taxi nhìn ra ngoài với vẻ tìm kiếm. Khi người này vô tình nhìn về hướng Sherlock, chiếc xe nổ máy và rời đi. Sherlock thông thuộc rõ các con đường tại thành phố anh sống. Anh cùng Watson chạy qua các lối đi tắt và đuổi theo chiếc taxi.

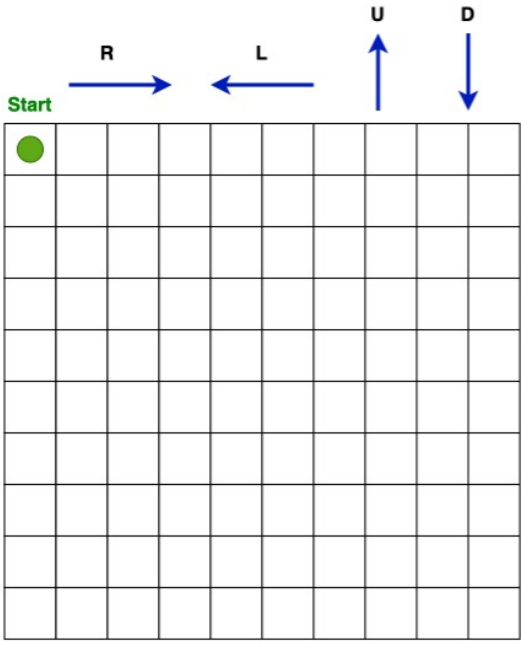
Sinh viên được yêu cầu viết hàm sau để mô tả lại quá trình này. Thông tin của hàm như sau:

- Tên hàm: **chaseTaxi**.
- Tham số đầu vào:

- `arr`: Chứa địa chỉ của phần tử đầu tiên của một mảng 2 chiều có kích thước cố định 100x100. Mỗi phần tử của mảng là một số nguyên. Mảng 2 chiều này biểu diễn bản đồ mà chiếc Taxi chạy.
- `points`: Chuỗi biểu diễn các điểm mà Sherlock dự kiến sẽ gặp chiếc taxi và chặn nó lại. Các điểm này được ngăn cách nhau bởi một ký tự ';' (dấu chấm phẩy). Mỗi điểm được biểu diễn dưới dạng:

  $$(p_i, p_j)$$

  với p_i, p_j lần lượt là chỉ số của điểm đó theo hàng và theo cột.
- `moves`: Chuỗi chứa các bước di chuyển của Taxi. Mỗi ký tự là một trong 4 chữ cái sau:
  - 'U': đi lên (up)
  - 'D': đi xuống (down)
  - 'L': qua trái (left)
  - 'R': qua phải (right)
- `outTimes`: Biến kiểu chuỗi được truyền tham khảo, dùng để trả về thời gian mà Sherlock và Watson di chuyển đến các điểm.
- `outCatchUps`: Biến kiểu chuỗi được truyền tham khảo, dùng để trả về kết quả về việc Sherlock và Watson có bắt kịp Taxi tại các điểm hay không.



- Yêu cầu hàm:

  1. Thông qua biến `arr` (cùng kích thước đã biết là 100x100) khởi tạo tất cả các phần

tử của mảng 2 chiều thành giá trị -9. Giá trị này biểu diễn tất cả các ô vuông đều chưa được Taxi đi qua.

2. Ban đầu, Taxi ở vị trí (0,0) như hình biểu diễn ở trên. Dựa vào biến `moves`, SV biểu diễn việc di chuyển của Taxi lần lượt qua các ô trên mảng 2 chiều tương ứng với từng ký tự đã được giải thích. Giả sử thời gian di chuyển giữa 2 ô liền kề nhau là 14 (đơn vị thời gian). Khi Taxi, ở vị trí (0,0), ta gán giá trị cho ô này là 0. Với mỗi bước khi Taxi đi đến một ô mới trên bản đồ, ta gán giá trị cho ô đó bằng giá trị của ô đã đứng trước đó cộng thêm 14. Trong lúc di chuyển, nếu Taxi đã đến một hàng/cột là biên của bản đồ, mà bước tiếp theo của `moves` là đi ra ngoài bản đồ, Taxi sẽ bỏ qua bước tiếp theo và đứng yên tại vị trí đó.
3. Với mỗi điểm trong chuỗi `points`, ta tính thời gian mà Sherlock và Watson sẽ đến được điểm đó. Thời gian di chuyển đến một điểm p_i bằng tổng thời gian để di chuyển đến điểm $p_i - 1$ và thời gian đi từ điểm $p_i - 1$ đến điểm p_i. Tuy nhiên, thời gian di chuyển đến điểm đầu tiên sẽ được tính bằng thời gian di chuyển từ vị trí (0,0) đến điểm đó. Vì Sherlock và Watson chạy bộ nên tốc độ di chuyển của 2 người sẽ chậm hơn Taxi, tốc độ này là 9 (đơn vị thời gian) khi đi qua 1 ô. Gọi d là khoảng cách Mahattan giữa 2 điểm, thời gian di chuyển giữa 2 điểm là: $9 * d$. Khoảng cách Mahattan giữa 2 điểm (x1,y1) và (x2,y2) được tính là:

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

- Kết quả trả về:
  - Hàm trả về giá trị `true` nếu Sherlock và Watson đuổi kịp Taxi. Ngược lại, trả về `false`.
  - `outTimes`: Chuỗi biểu diễn thời gian Sherlock và Watson di chuyển đến các điểm. Các thời gian được nối với nhau theo thứ tự của các điểm và ngăn cách bởi 1 ký tự ';' (dấu chấm phẩy). Nếu tại một điểm nào đó, 2 người đã bắt kịp Taxi thì các điểm sau đó, ta không cần tính thời gian di chuyển đến các điểm này. Ta sẽ thay nó bằng ký tự '-'.
  - `outCatchUps`: Chuỗi biểu diễn tại mỗi điểm Sherlock có bắt kịp Taxi hay không. Nếu có, ta biểu diễn bằng ký tự 'Y', nếu không, ta biểu diễn bằng ký tự 'N'. Nếu đây là một điểm sau điểm mà đã bắt kịp Taxi, ta biểu diễn bằng ký tự '-'. Các ký tự này sau đó được nối với nhau thành một chuỗi, mỗi ký tự được ngăn cách bởi 1 ký tự ';' (dấu chấm phẩy). Một điểm mà Sherlock bắt kịp Taxi nếu thời gian Sherlock đến được điểm đó không vượt quá thời gian mà Taxi đến được điểm đó.

Ví dụ 6: Đây là ví dụ về một giá trị của `points`:

- `"(20,30)-(70,90)"`: có 2 điểm Sherlock dự định sẽ bắt kịp Taxi. Điểm 1: chỉ số theo hàng là 20, chỉ số theo cột là 30. Điểm 2: chỉ số theo hàng là 70, chỉ số theo cột là 90.

Ví dụ 7: Đây là ví dụ về một giá trị của `moves`:

- `"RRRUDL"`
- Vị trí lần lượt thay đổi là:
  $(0,0) \xrightarrow{R} (0,1)$
  $\xrightarrow{R} (0,2)$
  $\xrightarrow{R} (0,3)$
  $\xrightarrow{U} (0,3)$ (đang ở biên trên của bản đồ nên không di chuyển lên trên)
  $\xrightarrow{D} (1,3)$
  $\xrightarrow{L} (1,2)$

3.3 Nhiệm vụ 3: Mật khẩu Laptop (2 điểm)

Sau khi đuổi theo chiếc Taxi, Sherlock và Watson quay lại căn hộ để nghỉ ngơi. Sherlock bị chú ý bởi chiếc Laptop tìm được trong hành lý. Bên ngoài hành lý có một chiếc thẻ có địa chỉ email của nạn nhân, anh đã thử dùng email này cho tên đăng nhập, còn mật khẩu thì anh thử tất cả những mật khẩu thông thường nhưng đều không được. Thanh tra Lestrade gọi điện đến nhắc Sherlock đã điều tra được thông điệp của nạn nhân để lại. Thông điệp này là tên của cô con gái đã mất cách đây vài năm của nạn nhân.

SV được yêu cầu viết hàm sau để mô tả lại quá trình này. Thông tin của hàm như sau:

- Tên hàm: **enterLaptop**.
- Tham số đầu vào:
  - `tag`: Chuỗi chứa tên của tập tin, tập tin này chứa thông tin của tấm thẻ bên ngoài hành lý.
  - `message`: Chuỗi chứa tên của con gái nạn nhân.
- Cấu trúc của tập tin **tag**:

Dòng	Nội dung
1	Email: <email của nạn nhân>
2	Address: <n3> THT Street

Với <n3> là chuỗi biểu diễn một số nguyên trong khoảng $[1, 20]$.

- Yêu cầu hàm: Tìm mật khẩu của laptop biết rằng mật khẩu này được tạo thành bằng cách lặp lại ghép tên của con gái nạn nhân <n3> lần (xem thêm ví dụ bên dưới).
- Kết quả trả về: Gọi mật khẩu của laptop tìm được là **pwdL**. Hàm sẽ trả về chuỗi bằng cách ghép email của nạn nhân và **pwdL**, ngăn cách 2 thành phần này bởi 1 ký tự ';' (dấu chấm phẩy): **<email của nạn nhân>;<pwdL>**.

Ví dụ 8: Dây là ví dụ về nội dung của tập tin **tag**:

Dòng	Nội dung
1	Email: pinky@cse.com
2	Address: 3 THT Street

Ví dụ 9: Với tên cô con gái là **Rachel** và <n3> = 3 thì mật khẩu của laptop là **Rachel-RachelRachel**.
Giải sử nội dung của tập tin **tag** giống với ví dụ trước. Hàm trả về giá trị: **"pinky@cse.com;RachelRachelRachel"**.

3.4 Tạm kết

Sau khi mở được laptop, trên Desktop của nạn nhân chỉ có một số phần mềm văn phòng cơ bản và 1 phần mềm định vị. Phần mềm này đã được cài đặt kết nối đến điện thoại của nạn nhân. May mắn là, tên đăng nhập cùng mật khẩu của phần mềm đều giống với thông tin này khi đăng nhập laptop. Phần mềm bắt đầu tìm kiếm, khu vực trên màn hình bắt đầu thu nhỏ phạm vi và hiển thị địa chỉ 221B phố Baker. Bà Hudson chạy lên báo một chiếc Taxi đậu đang đậu dưới nhà, tài xế nhờ bà chuyển lời: **"Taxi đặc biệt dành cho Sherlock Holmes."** Sherlock chợt hiểu ra mọi chuyện, anh bảo Watson ở lại tiếp tục tìm kiếm, anh cần ra ngoài hít thở một chút.

Sherlock sẽ làm gì để đối phó với tài xế taxi ngoài cửa, người mà khả năng cao chính là hung thủ của 4 vụ án tự tử. Liệu Watson có theo kịp Sherlock và hỗ trợ anh đánh bại tên tội phạm này. Chúng ta sẽ tiếp tục thực hiện các nhiệm vụ mới cùng Sherlock và Watson trong phần 3 của Bài tập lớn này.

Chúc các bạn làm Bài tập lớn vui vẻ!!!

4 Nộp bài

Sinh viên nộp 1 tập tin: **studyInPinkP2.h** trong site "Kỹ thuật lập trình (CO1027)_HK212_ALL"

Thời hạn nộp bài được công bố tại nơi nộp bài trong site nêu trên. Đến thời hạn nộp bài, đường liên kết sẽ tự động khoá nên sinh viên sẽ không thể nộp chậm. Để tránh các rủi ro có thể xảy ra vào thời điểm nộp bài, sinh viên PHẢI nộp bài trước thời hạn quy định ít nhất **một giờ**.

5 Xử lý gian lận

Bài tập lớn phải được sinh viên TỰ LÀM. Sinh viên sẽ bị coi là gian lận nếu:

- Có sự giống nhau bất thường giữa mã nguồn của các bài nộp. Trong trường hợp này, TẤT CẢ các bài nộp đều bị coi là gian lận. Do vậy sinh viên phải bảo vệ mã nguồn bài tập lớn của mình.
- Sinh viên không hiểu mã nguồn do chính mình viết, trừ những phần mã được cung cấp sẵn trong chương trình khởi tạo. Sinh viên có thể tham khảo từ bất kỳ nguồn tài liệu nào, tuy nhiên phải đảm bảo rằng mình hiểu rõ ý nghĩa của tất cả những dòng lệnh mà mình viết. Trong trường hợp không hiểu rõ mã nguồn của nơi mình tham khảo, sinh viên được đặc biệt cảnh báo là KHÔNG ĐƯỢC sử dụng mã nguồn này; thay vào đó nên sử dụng những gì đã được học để viết chương trình.
- Nộp nhầm bài của sinh viên khác trên tài khoản cá nhân của mình.

Trong trường hợp bị kết luận là gian lận, sinh viên sẽ bị điểm 0 cho toàn bộ môn học (không chỉ bài tập lớn).

KHÔNG CHẤP NHẬN BẤT KỲ GIẢI THÍCH NÀO VÀ KHÔNG CÓ BẤT KỲ NGOẠI LỆ NÀO!

Sau mỗi bài tập lớn được nộp, sẽ có một số sinh viên được gọi phỏng vấn ngẫu nhiên để chứng minh rằng bài tập lớn vừa được nộp là do chính mình làm.

6 Thay đổi so với phiên bản trước

Tài liệu

[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.

——————————HẾT——————————