TRƯỜNG ĐẠI HỌC BÁCH KHOA TP.HCM
Khoa Khoa học & Kỹ thuật Máy tính

BÀI TẬP LỚN
Hiện thực một công cụ hỗ trợ thanh toán QR

(Phiên bản v1.0)

1. Giới thiệu về công cụ hỗ trợ thanh toán QR

Hiện nay, xu hướng thanh toán trong nền kinh tế đang có sự chuyển dịch theo hướng sử dụng các phương tiện thanh toán không dùng tiền mặt nhiều hơn. Phương pháp này mang lại nhiều ưu điểm cho người dùng:

- **Không cần giữ một lượng lớn tiền mặt**: Người dùng sẽ không lo bị trộm cắp, tiền giả hay hoả hoạn.
- **Không cần gặp mặt trực tiếp**: Người dùng sẽ tiết kiệm được thời gian đi lại, từ đó giải quyết được nhiều công việc hơn.
- **Thông tin giao dịch cụ thể**: So với sử dụng tiền mặt trước đây, người dùng có nhu cầu sẽ phải ghi chép lại các chi tiêu trên giấy/sổ. Điều này có thể tốn thời gian, khó nhớ rõ từng khoản chi tiêu và nếu mất sổ ghi chép sẽ mất thông tin giao dịch. Thanh toán không tiền mặt thực hiện lưu trữ thông tin giao dịch một cách cụ thể và tự động. Người dùng khi có nhu cầu có thể xem lại giao dịch bất cứ lúc nào.
- **Nhận lãi từ ngân hàng**: Tiền nhàn rỗi trong tài khoản có thể được dùng để gửi tiết kiệm ngân hàng và mang lại thu nhập thêm (lãi suất) cho người dùng.

Nhận thấy các lợi ích trên, các ngân hàng bắt đầu triển khai các công nghệ hỗ trợ thanh toán không tiền mặt. Điển hình là hình thức thanh toán online qua E-Banking và Mobile Banking.

Với hai hình thức này, việc chuyển tiền thanh toán được thực hiện bằng cách nhập số tài khoản của người nhận và tên ngân hàng. Sau đó, hệ thống sẽ kiểm tra của số tài khoản này và hiển thị tên chủ tài khoản tương ứng để người gửi xác định đúng người nhận. Người gửi tiếp tục nhập số tiền và tin nhắn là có thể thanh toán đến người nhận.

Nhờ vào Cổng thanh toán quốc gia Napas[1], mọi giao dịch đều được diễn ra ngay lập tức giống như khi thanh toán tiền mặt. Mặc dù vậy, khi xét đến thanh toán trực tiếp (như thanh toán tại quầy thu ngân ở các cửa hàng và siêu thị), việc thanh toán online vẫn còn các hạn chế trong các bước thực hiện. Trước hết, người gửi cần có số tài khoản của người nhận để thực hiện, đây thường là dãy số dài và khó nhớ. Hơn nữa, người gửi có thể nhập sai mã số này dẫn đến việc nhập lại là khá mất công cho người gửi. Một số trường hợp nếu người gửi không kiểm tra kỹ sẽ chuyển nhầm tiền đến tài khoản khác.

Có lẽ, vẫn tồn tại cách thức có thể cải thiện hơn cách tương tác và cho phép người dùng thanh toán liên ngân hàng, không tiền mặt một cách dễ dàng, nhanh chóng hơn và ít sai sót từ việc nhập liệu của người dùng. Cách thức này có thể đạt được bằng ứng dụng quét mã QR.

Giải pháp thanh toán bằng mã QR hầu hết được triển khai như sau:

1. Người bán sẽ tạo một mã QR chứa thông tin tài khoản người nhận và đưa cho người mua.
2. Người mua khi thanh toán sẽ sử dụng điện thoại thông mình để quét mã QR trên và thực hiện việc chuyển tiền trên ứng dụng Mobile Banking của họ.

Di động thông minh ngày nay là một vật không thể thiếu đối với mỗi cá nhân. Vì vậy, thanh toán bằng mã QR là một giải pháp có thể thay thế việc thanh toán sử dụng tiền mặt hiện nay.

[1] CÔNG TY CỔ PHẦN THANH TOÁN QUỐC GIA VIỆT NAM - NAPAS

Điều này có thể được áp dụng cho các cửa hàng tiện lợi nhỏ lẻ khi giao dịch với khách hàng và giảm thiểu tiếp xúc. Từ đó, có thể giảm thiểu được nguy cơ lây lan dịch của đại dịch COVID-19 hiện nay. Đồng thời, đối với người dân, đặc biệt ở thành phố Hồ Chí Minh và các tỉnh miền Nam, việc mua sắm, thanh toán sẽ diễn ra một cách an toàn hơn. Môi trường phân phối sẽ không bị ách tắc, khó khăn, và giảm thiểu thiệt hại về kinh tế chung của các địa phương.

Trong khuôn khổ Bài tập lớn (BTL) này, chúng ta sẽ hiện thực một cách thức hỗ trợ việc chuyển tiền bằng mã QR giữa các ngân hàng khác nhau. Thông thường, khi sử dụng điện thoại thông minh quét mã QR, ta thường nhận được một liên kết, tổng quát hơn, ta nhận được một chuỗi ký tự. Trong ngữ cảnh của mã QR, chuỗi ký tự này được gọi là thông điệp (message). Thông điệp này là thứ có sẵn từ trước, sau đó sử dụng các phương pháp khác nhau để tạo nên mã QR. Các phương pháp này khá phức tạp và không phải mục tiêu chính của BTL này nên sẽ được bỏ qua. Như vậy, ta có thể từ thông điệp tạo ra mã QR và từ mã QR tìm lại được thông điệp. Một vấn đề tồn tại trong việc chuyển tiền liên ngân hàng là không thể hiểu được mã QR của nhau vì định dạng của mỗi thông điệp này là khác nhau ở mỗi ngân hàng. Từ đó, ứng dụng của ngân hàng không thể phân tích thông điệp thành các thông tin cần thiết và thực hiện chuyển khoản liên ngân hàng. Các thông tin này chính là các thông tin liên quan đến một tài khoản ngân hàng để xác định được tài khoản của người nhận.

Trong BTL này, sinh viên được yêu cầu hiện thực 2 lớp (class):

- **BankAccount**: Lớp này sẽ lưu trữ các thông tin quan trọng của một tài khoản ngân hàng và các phương thức liên quan.
- **BankQR**: Lớp này chứa các thông tin và các phương thức liên quan của một mã QR.

Các yêu cầu chi tiết sẽ được mô tả ở các Mục bên dưới.

2. Hiện thực lớp BankAccount

Class Account chứa các thông tin liên quan đến một tài khoản và các thao tác lên thông tin này. Các thông tin trong tài khoản:

- **Đầu số thẻ Việt Nam**: luôn là một chuỗi ký tự số “00020101021”
- **Số tài khoản ngân hàng**: một chuỗi toàn kí tự số, có độ tài từ 8 đến 15 kí tự tùy theo mỗi ngân hàng.

- **Tên chủ tài khoản**: chuỗi chứa tên của chủ tài khoản gồm các kí tự viết hoa hoặc viết thường có độ dài tối đa 30 kí tự. Tên của chủ tài khoản sẽ có một hoặc nhiều từ. Với mỗi từ, chữ cái đầu tiên phải là chữ cái viết hoa, các chữ cái sau chữ cái đầu tiên phải được viết thường, mỗi từ phải có ít nhất là một chữ cái viết hoa và một chữ cái viết thường. Trong chuỗi tên chủ tài khoản, các từ sẽ được viết liền nhau (tức là không có khoảng trắng giữa các từ).
    - Ví dụ về tên chủ tài khoản đúng:
        - "TranTrungTin"
    - Ví dụ về tên chủ tài khoản sai:
        - "Tran Trung Tin": sai vì có khoảng trắng giữa giữa các từ
        - "TRANTRUNGTIN": sai vì tất cả các chữ cái đều viết hoa, ta không thể tách thành từng từ sao cho chữ cái đầu viết hoa và các chữ cái sau viết thường.
        - "TranTrungT": sai vì từ cuối cùng không có chữ cái thường sau chữ cái viết hoa.
- **Tên ngân hàng**: chỉ được phép là 1 trong 3 chuỗi sau: "VCBKB", "OCBKB", "ACBKB"

Lưu ý: Với các yêu cầu về hiện thực trong BTL, nếu không nêu rõ tập tin cần hiện thực, sinh viên cần hiện thực vào file QRBank.cpp.

Yêu cầu 1.

Tìm hiểu class BankAccount trong file QRBank.hpp, class này dùng để chứa các thông tin về một tài khoản. Bạn cần đọc hiểu các thuộc tính (attributes) của class này, mỗi thuộc tính tương ứng với thông tin nào ở trên.

Yêu cầu 2.

Hiện thực 2 hàm Khởi tạo (Constructor) cho class BankAccount:

- Hàm khởi tạo không có tham số: khởi tạo các thuộc tính với giá trị "" (chuỗi rỗng).
- Hàm khởi tạo với 4 tham số: tên các tham số truyền vào tương ứng với thuộc tính cần gán giá trị. Giả sử rằng, các tham số được truyền vào ở hàm khởi tạo đều có giá trị hợp lệ.

Yêu cầu 3.

Hiện thực 4 cặp phương thức get/set cho 4 thuộc tính nêu trên. Ngoài ra, mỗi ngân hàng có quy định một độ dài cố định cho số tài khoản, sinh viên xem thêm Yêu cầu 5 về độ dài này.

Đối với các phương thức set, phương thức sẽ trả về kết quả là việc gán giá trị có thành công hay không. Nếu chuỗi truyền vào **KHÔNG** thoả các mô tả cho trường thông tin thì

phương thức sẽ không gán giá trị mới vào thuộc tính và trả về giá trị false. Ngược lại, phương thức sẽ gán giá trị mới cho thuộc tính và trả về true.

Yêu cầu 4.

Hai thông tin là số tài khoản và tên chủ tài khoản cần được mã hoá khi cần thiết để bảo mật thông tin cho người dùng. Việc thực hiện mã hoá được mô tả như sau:

a) Mã hoá cho số tài khoản

Số tài khoản được sẽ được mã hoá theo phương pháp gần giống với Mật mã Vigenère, sinh viên tự tìm hiểu thông tin của mật mã ngày tại Mật mã Vigenère – Wikipedia tiếng Việt. Điểm khác biệt là thay vì dùng 26 chữ cái in hoá (A-Z), ta sẽ dùng 10 chữ số (0-9) để xây dựng hình vuông Vigenère.

Phương pháp mã hoá Vigenère còn cần thêm 1 thông tin khác là từ khoá. Ta thực hiện xây dựng từ khoá như sau. Gọi s là tổng tất cả các chữ số của số tài khoản ban đầu. Đặt f = s % 3 + 3. Chia số tài khoản thành các đoạn bằng nhau, mỗi đoạn có độ dài là f, bắt đầu thực hiện chia từ phải qua trái. Trong trường hợp đoạn cuối cùng không đủ f chữ số, ta bù thêm vào phía trước một vài chữ số giống với chữ số đầu tiên cho đến khi đủ f chữ số thì dường. Gọi k là tổng tất cả các số thuộc các đoạn. Số k này sẽ được chọn làm từ khoá để mã hoá.

Ví dụ về cách tìm từ khoá:

Số tài khoản	2129082021
Tính s	(2+1+2+9+0+8+2+0+2+1)=27
Tính f	(27 % 3) + 3 = 3
Chia số tài khoản thành từng đoạn có độ dài là f=3	|2|129|082|021|
Đoạn đầu tiên chưa đủ 3 chữ số, ta bù vào đầu đoạn hai chữ số 2 (là chữ số đầu tiên của chuỗi)	|222|129|082|021|
Tính k	k = 222 + 129 + 82 + 21 = 454

Yêu cầu hiện thực hàm **getEncodedAccountNumber**:

`std::string getEncodedAccountNumber() const;`	
Input	+ Không
Output	+ Trả về: số tài khoản được mã hoá như mô tả trên.

b) Mã hoá cho tên chủ tài khoản

Việc mã hoá cho tên chủ tài khoản được thực hiện bằng Mã hoá Caesar, sinh viên tự tìm hiểu thông tin của mật mã này tại Mật mã Caesar – Wikipedia tiếng Việt. Mã hoá Caesar cần có thông tin về độ dời (ký hiệu là d). Gọi s là tổng tất cả các chữ số của số tài khoản sau khi được mã hoá. Gọi p là số nguyên tố nhỏ nhất không nhỏ hơn s. Khi đó, d được tính là:

$$d = p \% 52$$

Về bảng chữ cái được sử dụng, hai bảng chữ cái hoa và thường được ghép theo từng cặp và liên tiếp nhau như sau:

A	a	B	b	C	c	…	Z	z

Ví dụ về mã hoá chuỗi:

Độ dời d sau tính được là 2	
Chuỗi thô (chuỗi ban đầu)	ZzzBaaCzab
Chuỗi mã hoá	AaaCbbDabc

Yêu cầu hiện thực hàm **getEncodedAccountName**:

`std::string getEncodedAccountName() const;`	
Input	+ Không
Output	+ Trả về: tên tài khoản được mã hoá như mô tả trên.

Yêu cầu 5.

Trong yêu cầu này, sinh viên cần xây dựng thông điệp cho mã QR dựa vào các thông tin của tài khoản như trên, ngoài ra còn cần có thêm thông tin về thời điểm tạo thông điệp QR. Đây là một chuỗi có độ dài tối đa là 8 ký tự và có định dạng phụ thuộc vào từng ngân hàng. Chi tiết về định dạng của từng ngân hàng được mô tả ở các phần bên dưới

- **Ngân hàng VCBKB**

  Ngân hàng VCBBK quy định thông điệp QR của họ như sau:

  Đầu số thẻ VNTên ngân hàngSố tài khoảnTên chủ tài khoảnThời điểm tạo thông điệp QR

  Trong đó:

  - **Tên ngân hàng**: "VCBKB"
  - **Số tài khoản**: Là một chuỗi kí tự số có độ dài 13 kí tự
  - **Thời điểm tạo thông điệp QR**: Là một chuỗi ngày, tháng và năm theo format ddmmyyyy. Ví dụ: 22/08/2021 sẽ được lưu thành "22082021"

Một số ví dụ cho thông điệp QR của VCBKB:

- 00020101021VCBKB0271001142475TranMinhHoang22082021
- 00020101021VCBKB0491009260891NguyenVanBa26082021
- 00020101021VCBKB0882003729910DangVanThanh18072020

- **Ngân hàng OCBKB**

Ngân hàng OCBKB quy định thông điệp QR của họ như sau:

Đầu số thẻ VNThời điểm tạo thông điệp QRTên ngân hàngSố tài khoảnTên chủ tài khoản

Trong đó:

- **Tên ngân hàng**: "OCBKB"
- **Số tài khoản**: Là một chuỗi kí tự số có độ dài 15 kí tự
- **Thời điểm tạo QR**: Là một chuỗi ngày, tháng và năm theo format **mmddyyyy**. Ví dụ: 22/08/2021 sẽ được lưu thành "08222021"

Một số ví dụ cho thông điệp QR của OCBKB:

- 000201010210822202100CBKB012100002512112VoThiQuynhNhu
- 000201010210731202100CBKB013900001913490NguyenThinhToan
- 000201010211220202000CBKB012100000801332HoangCongAnh

- **Ngân hàng ACBKB**

Ngân hàng ACBKB quy định thông điệp QR của họ như sau:

Đầu số thẻ VNTên chủ tài khoảnSố tài khoảnThời điểm tạo thông điệp QRTên ngân hàng

Trong đó:

- **Tên ngân hàng**: "ACBKB"
- **Số tài khoản**: Là một chuỗi kí tự số có độ dài 8 kí tự
- **Thời điểm tạo thông điệp QR**: Là một chuỗi ngày, tháng và năm theo format yymmdd (yy là 2 số cuối của năm). Ví dụ: 22/08/2021 sẽ được lưu thành "210822".

Một số ví dụ cho thông điệp QR của ACBKB:

- 00020101021TrinhThiKhanhDuyen29136412200101ACBKB
- 00020101021HuynhNgocPhu14226317200706ACBKB
- 00020101021DangVanBinh15883012201130ACBKB

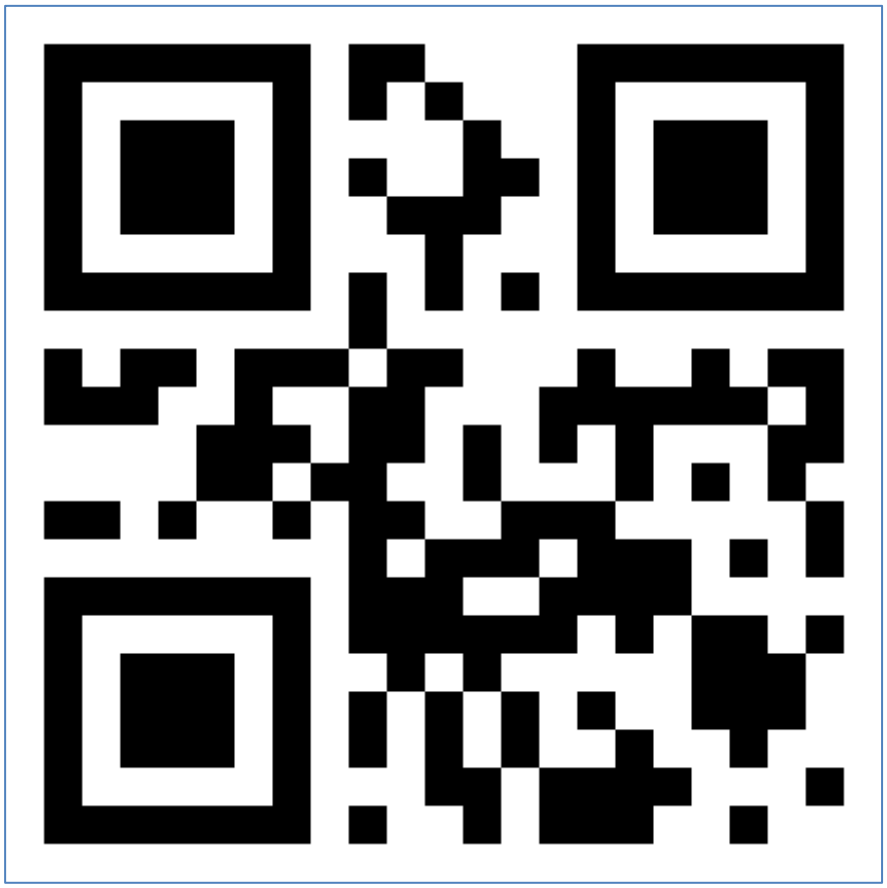
Hiện thực hàm **buildMessageForQR** trả về thông điệp được xây dựng từ thông tin tài khoản với các mô tả như trên.

`std::string buildMessageForQR(const std::string & createdAt) const;`	
Input	+ createdAt: thời điểm thông điệp được tạo, định dạng của thời điểm luôn là dd-mm-yyyy
Output	+ Trả về: chuỗi biểu diễn thông điệp cho mã QR ứng với mỗi ngân hàng theo mô tả trên.
Mô tả	- sinh viên cần chuyển giá trị của createdAt về đúng định dạng cho ***Thời điểm tạo QR*** của từng ngân hàng như đã được mô tả. - Để đơn giản, các ví dụ trên mới chỉ hiển thị số tài khoản và tên chủ tài khoản thẻ ở dạng chưa mã hoá. Tuy nhiên, 2 thông tin này phải được mã hoá (như mô tả ở Yêu cầu 4) trước khi đưa vào thông điệp. Tham khảo ô bên dưới về một ví dụ đúng cho việc xây dựng thông điệp của hàm này với 2 thông tin trên được mã hoá.
Ví dụ	Một tài khoản có các thông tin sau - Tên ngân hàng: **VCBKB** - Đầu số thẻ Việt Nam: **00020101021** - Tên chủ tài khoản: **VuVanTien** - Số tài khoản: **1234567890123** Thời điểm tạo thông điệp QR là: **31-08-2021**
	Ta được kết quả như sau: - Tên chủ tài khoản sau khi mã hoá: **cCcIVaQMV** - Số tài khoản sau khi mã hoá: **2159659309374** - Hàm **buildMessageForQR** trả về thông điệp QR: **00020101021VCBKB2159659309374cCcIVaQMV31082021**

3. Hiện thực lớp BankQR

Class BankQR chứa thông tin của một mã QR, class cũng cung cấp các phương thức cần thiết để làm việc với mã QR. Trước tiên ta tìm hiểu các thông tin của một mã QR:

- Thông điệp để tạo nên mã QR.
- Kích thước của mã QR (mã QR có hình vuông nên chỉ cần 1 giá trị để biển diễn kích thước)
- Một mảng 2 chiều lưu trữ dữ liệu của mã QR. Ta xem mã QR là một mảng 2 chiều mã mỗi phần tử trong mảng là 1 điểm ảnh (pixel, với QR mỗi điểm này còn được gọi là module). Một pixel chỉ có thể là 1 trong 2 màu đen hoặc trắng.



Yêu cầu 6.

Trong file QRBank.hpp, tìm hiểu class BankQR. Trong access modifier private, bạn hãy tự đề xuất và điền vào các thuộc tính cho class này. Sinh viên cần lưu ý, các yêu cầu sau yêu cầu này sẽ hiện thực các phương thức cho class BankQR. Ở yêu cầu này, SV có thể đề xuất bất kỳ thuộc tính nào, nhưng, cần đảm bảo các phương thức sau hoạt động đúng như mô tả. Nó sẽ được dùng để chạy các testcases chấm điểm cho BTL này. SV có thể ghi các thuộc tính trước, sau đó, trong lúc thực hiện các yêu cầu sau thì điều chỉnh lại nếu cần.

Yêu cầu 7.

Hiện thực hàm Khởi tạo cho class BankQR.

`BankQR(string message)`	
Input	+ message: thông điệp để tạo mã QR
Output	+ Trả về: không
Mô tả	Khởi tạo các thuộc tính với các giá trị phù hợp. Việc chuyển đổi từ thông điệp sang mã QR đã được hiện thực sẵn trong hàm encodeQR ở

	đường dẫn helper/helper.hpp.

Yêu cầu 8.

Hiện thực hàm Huỷ (Destructor) cho class BankQR. Hàm này phải thực hiện thu hồi bất kỳ vùng nhớ Heap nào mà hàm khởi tạo cấp phát.

`~BankQR()`	
Input	+ Không
Output	+ Trả về: không
Mô tả	Hàm cần thực hiện thu hồi bất kỳ vùng nhớ Heap nào mà hàm khởi tạo cấp phát.

Yêu cầu 9.

Hiện thực phương thức **saveToPNG** để ghi mã QR vào một ảnh PNG.

`bool saveToPNG(std::string filename) const;`	
Input	+ filename: tên file ghi mã QR vào (kết thúc bằng ".png").
Output	+ Trả về: true nếu việc ghi file là thành công, false nếu ngược lại. Các bạn xem và sử dụng hàm **writeQRToPNG** ở đường dẫn helper/helper.hpp để ghi QR vào file ảnh PNG.

Sau khi hoàn thành yêu cầu này, các bạn có thể dùng điện thoại thông minh để quét mã được sinh ra và xem xét thông điệp được hiển thị có phải là thông điệp bạn đã truyền vào hàm khởi tạo hay không.

Yêu cầu 10.

Hiện thực phương thức **toString**, trả về một chuỗi biểu diễn của mã QR theo mô tả sau.

`string toString() const`	
Input	+ Không
Output	+ Trả về một chuỗi biểu diễn mã QR.
Mô tả	- Với mỗi pixel, nếu pixel có màu trắng thì biểu diễn bằng chuỗi có 2 khoảng trắng (space) " ", nếu pixel có màu đen thì biểu diễn bằng chuỗi có 2 dấu thăng *"##"*. - Chuỗi sau khi in ra màn hình phải đảm bảo không có ký tự khoảng trắng ở cuối mỗi hàng, và kết thúc hàng cuối không được có thêm ký tự xuống hàng.
Ví dụ	- Thông điệp để khởi tạo mã lớp BankQR (lấy chuỗi trong dấu nháy kép): **"Hello world!!!"**

	Hàm trả về chuỗi như bên dưới

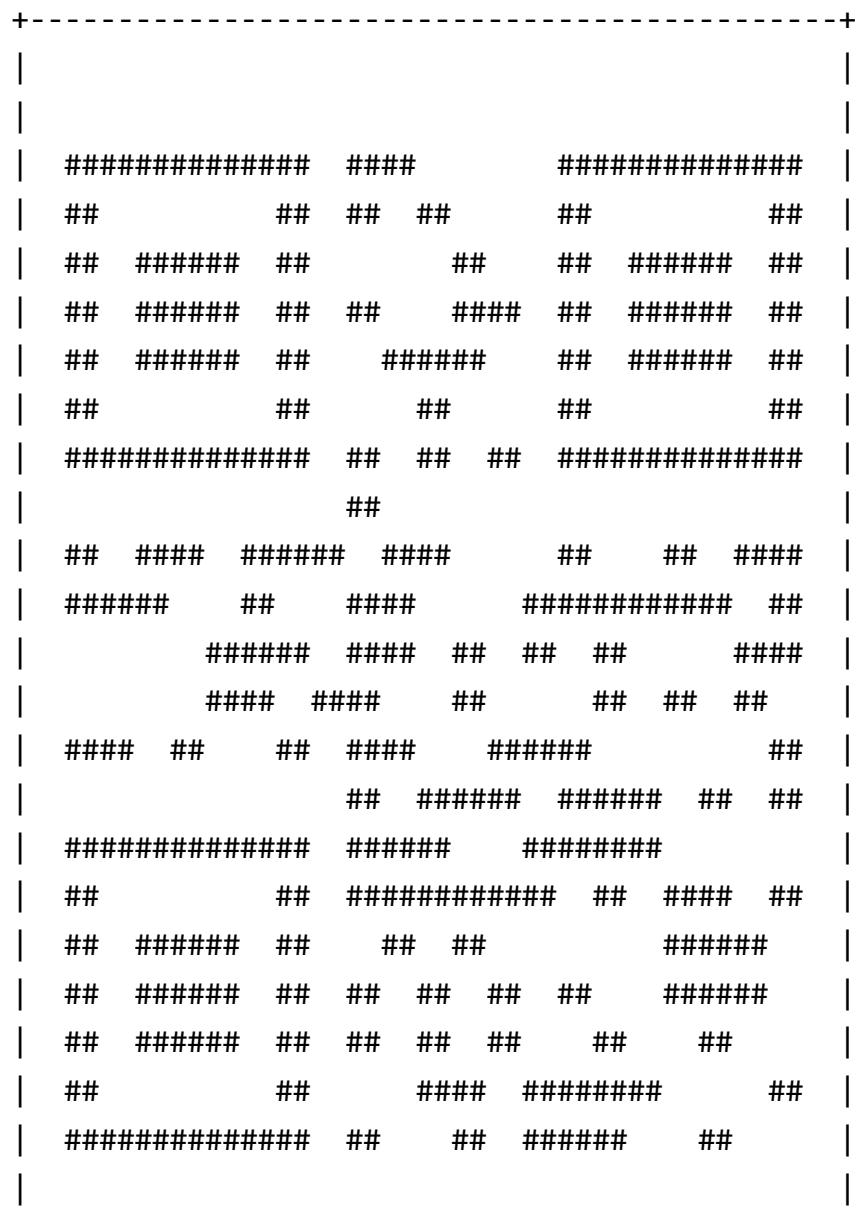
```
##############  ####        ##############
##          ##  ##  ##      ##          ##
##  ######  ##      ##      ##  ######  ##
##  ######  ##  ##    ####  ##  ######  ##
##  ######  ##    ######    ##  ######  ##
##          ##      ##      ##          ##
##############  ##  ##  ##  ##############
                ##
##  ####  ######  ####      ##    ##  ####
######    ##    ####      ############  ##
        ######  ####  ##  ##  ##      ####
        ####  ####    ##      ##  ##  ##
####  ##    ##  ####    ######          ##
                ##  ######  ######  ##  ##
##############  ######    ########
##          ##  ############  ##  ####  ##
##  ######  ##    ##  ##          ######
##  ######  ##  ##  ##  ##  ##    ######
##  ######  ##  ##  ##  ##    ##    ##
##          ##      ####  ########      ##
##############  ##    ##  ######    ##
```

Yêu cầu 11.

Hiện thực phương thức **toString(int margin)**, trả về một chuỗi biểu diễn của mã QR theo mô tả sau.

`string toString(int margin) const`	
Input	+ margin: số nguyên biểu diễn khoảng cách từ các cạnh của QR đến 4 viền xung quanh. margin có giá trị >= 0. Testcase cung cấp sẽ đảm bảo điều kiện này nên SV không cần kiểm tra.
Output	+ Trả về: một chuỗi biểu diễn mã QR.
Mô tả	Chuỗi biểu diễn mã QR giống như mô tả ở Yêu cầu 10. Tuy nhiên, xung quanh mã QR có một khung viền, khoảng cách từ khung viền đến cạnh của mã QR là margin (xem thêm phần *Ví dụ*). - ***Viền***: Với viền trên và viền dưới, mỗi ký tự được sử dụng là

	‘-‘. Với viền trái và viền phải, mỗi ký tự được sử dụng là ‘|’. Với các ký tự tại 4 góc, ký tự được sử dụng là ‘+’. - ***Khoảng cách với viền***: viền trên và cạnh trên của QR cách nhau margin dòng; viền dưới và cạnh dưới của QR cách nhau margin dòng; viền trái và cạnh trái của QR, viền phải và cạnh phải của QR đều cách nhau margin cột, với một cột có độ dài là 1 khoảng trắng.
Ví dụ	- Thông điệp để khởi tạo mã lớp BankQR (chuỗi trong dấu nháy kép: **"Hello world!!!"** - margin = 2 Hàm trả về chuỗi như bên dưới



	| | +----------+

Yêu cầu 12.

Hiện thực phương thức **saveStringTo** để ghi chuỗi biểu diễn mã QR vào một file.

`bool saveStringTo(std::string filename, int margin=0) const;`	
Input	+ filename: tên file mà chuỗi biểu diễn ghi vào. + margin: số nguyên biểu diễn khoảng cách từ các cạnh của QR đến 4 viền xung quanh. margin có giá trị >= 0. Testcase cung cấp sẽ đảm bảo điều kiện này.
Output	+ Trả về true nếu việc ghi file là thành công, false nếu ngược lại.
Mô tả	Ghi vào file chuỗi biểu diễn giống như mô tả ở Yêu cầu 11.

Yêu cầu 13.

Hiện thực phương thức **decodePNG** giải mã một file ảnh QR ở định dạng PNG sang thông điệp ban đầu được dùng để tạo ra mã QR này.

`static std::string decodePNG(std::string filename);`	
Input	+ filename: tên file ảnh QR cần đọc (kết thúc bằng ".png")
Output	+ Trả về thông điệp dùng để tạo ra mã QR này.

Ở yêu cầu này, các bạn cần tự tìm hiểu class QRCodeDetector của thư viện OpenCV (version 4.5.2). Sau đó sử dụng class này để hiện thực phương thức trên.

Lưu ý: Đối với việc cài đặt thư viện OpenCV trên hệ điều hành *Windows*, phần lớn các nguồn sẽ hướng dẫn các bạn cài đặt sử dụng Visual Studio. Tài liệu này đề xuất một cách khác để các bạn tham khảo: tìm hiểu MSYS2 và cài đặt thư viện OpenCV bằng MSYS2. Các bạn đều có thể gặp khó khăn khi cài đặt thư viện OpenCV dù dùng cách nào, hãy cố gắng tìm kiếm thông tin trên Internet và giải quyết các rắc rối này!!!

Yêu cầu 14.

Hiện thực hàm **decodeInfoOfMessage** trích xuất 4 thông tin từ một thông điệp, hàm này trả về một chuỗi biểu diễn in 4 thông tin này theo mô tả sau.

```
static std::string decodeInfoOfMessage(const std::string &
                                       message);
```

Input	+ message: thông điệp cần trích xuất thông tin
Output	+ Trả về chuỗi biểu diễn thông tin như trong phần *Mô tả*.
Mô tả	Ngan hang: **<Tên ngân hàng>** So tai khoan: **<Số tài khoản đã mã hoá>** Ten chu tai khoan: **<Tên chủ tài khoản đã mã hoá>** Thoi diem tao thong diep QR: **<Thời điểm tạo thông điệp QR>**
	Lưu ý: - **Tên ngân hàng**, **Số tài khoản đã mã hoá**, **Tên chủ tài khoản đã mã hoá** sẽ giống với Mục 2. - **Thời điểm tạo thông điệp QR** có định dạng là dd/mm/yyyy (ví dụ: 29/08/2021) - Nếu thời điểm tạo thông điệp QR chỉ có 2 chữ số cuối cho phần biểu diễn năm thì 2 chữ số đầu của năm sẽ là 20. Ví dụ: Thời điểm trong thông điệp là "220821", thì chuỗi biểu diễn sẽ là "21/08/2022". - Sau mỗi dấu hai chấm (:) là một khoảng trắng ngăn cách.
Ví dụ	- Thông điệp QR cần giải mã là: **00020101021VCBKB2159659309374cCcIVaQMV01092021** Hàm trả về chuỗi như bên dưới

	`Ngan hang: VCBKB`
	`So tai khoan: 2159659309374`
	`Ten chu tai khoan: cCcIVaQMV`
	`Thoi diem tao thong diep QR: 01/09/2021`

Yêu cầu 15.

Mặc dù mỗi ngân hàng đều có mã QR của riêng mình, hầu hết các ứng dụng mobile banking của một ngân hàng chỉ nhận diện được mã QR của chính ngân hàng đó (không đọc được định dạng thông điệp QR của các ngân hàng khác). Để khắc phúc tình trạng trên, SV hãy hiện thực hàm **convert** thực hiện việc chuyển mã QR ở định dạng ban đầu sang mã QR có định dạng của ngân hàng yêu cầu.

`static BankQR convert(const BankQR & inQR, const std::string & targetBank);`	
Input	+ inQR: mã QR cần chuyển đổi + targetBank: biểu diễn tên của một ngân hàng, mà mã QR này cần phải đổi sang định dạng của ngân hàng đó
Output	+ Mã QR ở định dạng của ngân hàng đích.

4. Một số yêu cầu và hướng dẫn khác

- Sinh viên không được phép thêm bất cứ thư viện nào khác ngoài các thư viện đã có sẵn.
- Đối với file QRBank.hpp, sinh viên không được thay đổi **4** dòng đầu tiên. Đối với file QRBank.cpp, sinh viên không được thay đổi **9** dòng đầu tiên.
- Về cách biên dịch chương trình: hàm cung cấp sẵn trong helper cần link thư viện **stdc++fs** và sử dụng ổn định ở phiên bản **c++17**; các hàm hiện thực sử dụng OpenCV cần link 3 thư viện **opencv_objdetect**, **opencv_imgcodecs**, **opencv_core**, và cần thêm đường dẫn đến thư mục chứa header của OpenCV. Giả sử file chương trình chứa hàm main là main.cpp đặt ở thư mục gốc của file giải nén. Câu lệnh biên dịch sẽ có dạng như sau:

```
g++ -std=c++17 -o main -lstdc++fs -lopencv_objdetect -
lopencv_imgcodecs -lopencv_core -I <đường dẫn đến header
OpenCV> main.cpp helper/*.cpp helper/libs/*.cpp -I
helper/*.hpp -I helper/libs/*.hpp
```

Sau khi chạy câu lệnh trên, ta có được file thực thi main và có thể chạy chương trình bằng lệnh: **`./main`**

5. Nộp bài

Sinh viên download file *Do_an_KTLT_assigment.zip* từ trang Web của môn học. Khi giải nén file này, sẽ có được các file sau:

[Do_an_KTLT]Assignment.pdf		File mô tả nội dung bài tập lớn
Thư mục *initial_code*	*QRBank.hpp*	File header khai báo cho 2 lớp BankAccount và BankQR
	QRBank.cpp	File để điền phần hiện thực cho 2 lớp BankAccount và BankQR
	Thư mục *helper*	Chứa hàm được cung cấp sẵn và một số thư viện cần thiết

Khi nộp bài, sinh viên nộp bài trên site e-Learning của môn học. Sinh viên nộp 2 tập tin là **`QRBank.hpp`** và **`QRBank.cpp`**.

5.1. Thời hạn nộp bài

Thời gian nộp bài sẽ được công bố trong site e-Learning của môn học. Đến thời hạn nộp bài, nơi nộp bài sẽ tự động bị 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**. Nếu sinh viên nộp bài trễ, điểm tối đa của bài tập lớn sẽ là **0 điểm**. KHÔNG nhận bài được gửi qua e-mail hoặc bất kỳ hình thức nào khác.

5.2. 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.

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.

-HẾT-