

Assignment

MÔN: CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

MÃ MÔN: 504008

Đề bài: Quản lý danh sách sinh viên

(Sinh viên đọc kỹ tất cả hướng dẫn trước khi làm bài)

I. Giới thiệu bài toán

Một đơn vị cần quản lý danh sách sinh viên của đơn vị mình. Sinh viên sẽ hiện được một phần mềm quản lý danh sách sinh viên. Tuy nhiên yêu cầu của đơn vị này là phải dùng Danh sách liên kết (Linked List) để lưu trữ.

Các chức năng được yêu cầu phải thực hiện là:

- Thêm sinh viên vào danh sách và đảm bảo danh sách phải có thứ tự theo mã số sinh viên.
- Xóa sinh viên theo mã số sinh viên.
- Sửa thông tin sinh viên.
- Tìm sinh viên trong danh sách theo yêu cầu.
- Có chức năng undo.

Cụ thể các yêu cầu này sẽ được mô tả chi tiết ở các phần bên dưới.

II. Tài nguyên cung cấp

Source code được cung cấp sẵn bao gồm các file:

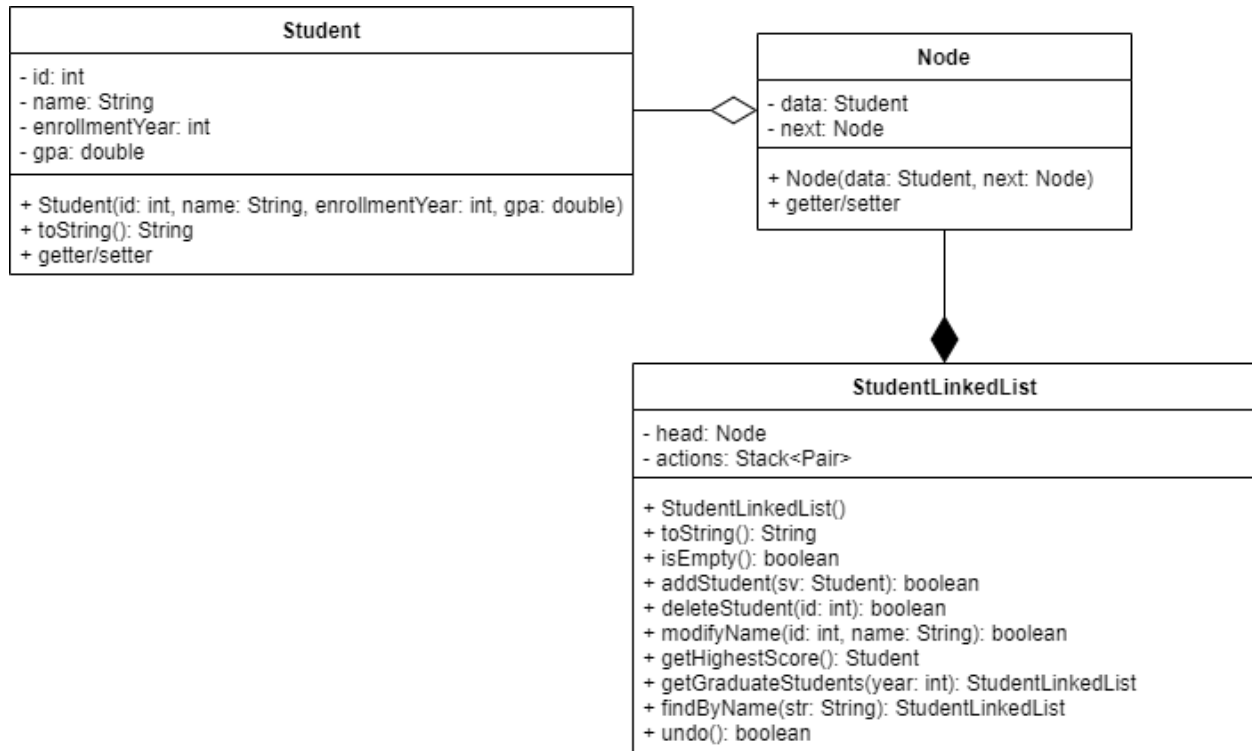
- File đầu vào và kết quả mong muốn:
 - o Folder *input* chứa file *Students.txt*, đây là file chứa thông tin sinh viên, danh sách này sẽ được dùng từ Yêu cầu 1 đến Yêu cầu 6 để kiểm thử các phương thức của sinh viên định nghĩa.
 - o Folder *output* gồm: 7 file *Req1.txt*, *Req2.txt*, *Req3.txt*, *Req4.txt*, *Req5.txt*, *Req6.txt*, *Req7.txt* chứa kết quả mẫu của các yêu cầu trong bài.
- File mã nguồn:
 - o *Main.java*: tạo đối tượng và gọi các phương thức sinh viên sẽ định nghĩa, trong file này định nghĩa sẵn câu lệnh kiểm thử từng yêu cầu của đề bài.
 - o *Student.java*: chứa lớp **Student** được định nghĩa sẵn để tạo đối tượng sinh viên. File này sinh viên không được chỉnh sửa.
 - o *Node.java*: chứa lớp **Node** được định nghĩa sẵn để tạo các nút của danh sách liên kết. File này sinh viên không được chỉnh sửa.

- *Pair.java*: chứa lớp **Pair** được định nghĩa sẵn gọi ý cho Yêu cầu 7. File này sinh viên không được chỉnh sửa. Sinh viên sẽ dùng lớp này để hiện thực Yêu cầu 7.
- *Helper.java*: chứa lớp **Helper** được định nghĩa sẵn phương thức đọc file và ghi file được gọi trong phương thức main. File này sinh viên không được chỉnh sửa.
- *StudentLinkedList.java*: chứa lớp **StudentLinkedList** được định nghĩa sẵn thuộc tính *head* là head của danh sách liên kết, *Stack action* dùng cho Yêu cầu 7, phương thức bổ trợ và một số phương thức trống. Sinh viên sẽ hiện thực thêm vào các phương thức còn trống trong file này và không chỉnh sửa phương thức có sẵn.

III. Trình tự thực hiện bài

- Sinh viên tải file tài nguyên được cung cấp sẵn và giải nén.
- Đọc kĩ đề và đọc hiểu các file được cung cấp sẵn để hiểu rõ cách hoạt động của các phương thức cung cấp sẵn.
- Trong lớp **StudentLinkedList**, sinh viên hiện thực theo thứ tự từ Yêu cầu 1. Sinh viên phải hiện thực được Yêu cầu 1 đúng thì mới có thể làm tiếp các yêu cầu còn lại.
- Sau khi hiện thực các phương thức trong lớp **StudentLinkedList**, sinh viên biên dịch và chạy với phương thức **main** trong file *Main.java* đã gọi sẵn các phương thức. Sinh viên thực hiện các yêu cầu ở mục dưới và so sánh kết quả của mình với kết quả trong folder output đã được cung cấp sẵn.
- **Đối với các yêu cầu sinh viên không làm được vui lòng không xóa và phải đảm bảo chương trình chạy được với phương thức main trong Main.java được cung cấp sẵn.**
- Đường link Google Drive gửi đề này cho sinh viên sẽ chứa kèm một file *version.txt*, sinh viên nên thường xuyên vào xem file này, nếu thấy có nội dung mới thì nên đọc kĩ mô tả và tải lại đề mới nhất. File này được dùng để thông báo nội dung chỉnh sửa và ngày chỉnh sửa trong trường hợp đề bị sai hoặc lỗi.

IV. Mô tả các lớp trong bài



- Giải thích một số thuộc tính và phương thức như sau:
 - **Lớp Student**
 - *id*: mã số sinh viên.
 - *name*: họ và tên sinh viên.
 - *enrollmentYear*: năm nhập học.
 - *gpa*: điểm tích lũy.
 - **toString()**: trả về chuỗi thể hiện cho đối tượng **Student**.
 - Các phương thức get/set để thao tác với dữ liệu.
 - **Lớp Node**
 - *data*: lưu trữ đối tượng sinh viên.
 - *next*: nút tiếp theo trong danh sách liên kết.
 - Các phương thức get/set để thao tác với dữ liệu.
 - **Lớp Helper**
 - **readFile(String path)**: trả về danh sách liên kết sinh viên đọc từ file.
 - **writeFile(String path, E data)**: ghi file dữ liệu *data* truyền vào.
 - **Lớp Pair** – dùng cho Yêu cầu 7
 - *node*: nút đang xét
 - *action*: thao tác
 - **toString()**: trả về chuỗi thể hiện cho đối tượng **Pair**.
 - Các phương thức get/set để thao tác với dữ liệu.
 - **Lớp StudentLinkedList**

- *head*: nút đầu tiên trong danh sách.
- *actions*: Stack chứa các thao tác cho Yêu cầu 7.
- **toString(), isEmpty()**: là các phương thức phụ trợ. Sinh viên có thể dùng phương thức này nếu cần. Sinh viên không chỉnh sửa các phương thức này.

V. Mô tả file input và file output

- File input có tên *Students.txt* chứa danh sách sinh viên với thuộc tính của 01 nhân viên được cách nhau bằng dấu “-” theo định dạng:

Mã số sinh viên - Họ và tên - Năm nhập học - Điểm tích lũy

```
1 51301161 - Le Minh Phuong - 2013 - 7.5
2 51502010 - Nguyen An Khang - 2015 - 5.6
3 51201010 - Ho Dai Phat Tai - 2012 - 9.2
```

Trong source code được cung cấp sẵn đã có định nghĩa phương thức để đọc file này. Sinh viên không tự ý chỉnh sửa.

- File output gồm có 7 file ứng với kết quả mẫu cho các yêu cầu:
 - *Req1.txt*: kết quả danh sách sinh viên đọc từ file.
 - *Req2.txt*: kết quả danh sách sinh viên thực hiện 03 lệnh xóa sinh viên.
 - *Req3.txt*: kết quả danh sách sinh viên thực hiện 01 lệnh sửa họ tên sinh viên.
 - *Req4.txt*: kết quả sinh viên đầu tiên có điểm tích lũy cao nhất.
 - *Req5.txt*: kết quả danh sách sinh viên đã tốt nghiệp ở năm 2018.
 - *Req6.txt*: kết quả danh sách sinh viên có tên Quang.
 - *Req7.txt*: kết quả danh sách sinh viên thực hiện các câu lệnh thêm, xóa, undo.
- Tất cả đối tượng sinh viên khi ghi file đều có định dạng theo phương thức **toString()** của lớp **Student**. Do đó, sinh viên không được phép sửa các phương thức **toString()**.

Lưu ý:

- Sinh viên có thể tự thêm dữ liệu vào file input để thử nhiều trường hợp khác nhau nhưng lưu ý thêm dữ liệu phải đúng định dạng đã được nêu bên trên.
- Sinh viên nên đọc kỹ phương thức **main** để xác định hướng hiện thực các lớp và các phương thức theo thứ tự..
- Sinh viên có thể thêm phương thức mới để phục vụ bài làm tuy nhiên bài làm của sinh viên phải chạy được với file *Main.java* đã được cung cấp sẵn. Các phương thức thêm mới phải nằm trong các file nộp và không gây ảnh hưởng các phương thức có sẵn.
- Sinh viên tuyệt đối không đặt các đường dẫn cứng trên máy khi định nghĩa phương thức liên quan đến đọc file. Nếu sinh viên tự ý đặt đường dẫn cứng dẫn đến quá trình chấm không đọc được file để chấm thì tương đương với bài làm biên dịch lỗi.
- **Tuyệt đối không chỉnh sửa tên của các phương thức đã có sẵn (tuân theo đúng sơ đồ lớp phía trên).**

VI. Hướng dẫn chạy phương thức main

- Phương thức **main** được cung cấp sẵn trong file *Main.java* được tạo sẵn các đối tượng danh sách và gọi các phương thức để kiểm thử các yêu cầu.
- Để kiểm thử các yêu cầu, sinh viên phải định nghĩa các phương thức đúng theo yêu cầu, sau khi biên dịch thành công, sinh viên gọi theo cú pháp để in ra kết quả của yêu cầu tương ứng:

java Main X

- Với $X = \{1, 2, 3, 4, 5, 6, 7\}$ sẽ thực hiện kiểm thử yêu cầu tương ứng.
Ví dụ, sinh viên muốn gọi để kiểm thử Yêu cầu 1, sinh viên nhập *java Main 1*, kết quả sẽ được ghi ra file *Req1.txt*.
- Sau khi đã kiểm thử thành công với **main** cung cấp sẵn, sinh viên có thể chỉnh sửa một số thao tác trong phương thức **main** để kiểm thử các phương thức đã định nghĩa tuy nhiên đảm bảo bài làm của sinh viên **phải chạy được trên phương thức main ban đầu đã cung cấp sẵn**.

VII. Yêu cầu

Sinh viên không tự ý thêm thư viện ngoài, chỉ dùng những thư viện có sẵn trong các file nhận được.

Sinh viên thực hiện bài tập lớn trên Java 11 hoặc Java 8. Sinh viên không được dùng kiểu dữ liệu *var*. Bài làm của sinh viên sẽ được chấm trên Java 11, sinh viên tự chịu trách nhiệm tất cả các lỗi phát sinh nếu dùng các phiên bản Java khác.

1. YÊU CẦU 1 (2 điểm)

Sinh viên hiện thực phương thức **public boolean addStudent(Student sv)** thêm một nút lưu trữ đối tượng **Student sv** vào danh sách liên kết. Trong quá trình thêm phải đảm bảo danh sách có thứ tự mã số sinh viên từ bé đến lớn. Nếu thêm thành công thì trả về *true*, trường hợp đối tượng **Student** thêm vào có mã số sinh viên trùng với mã số sinh viên của bất kì sinh viên nào đang có sẵn thì không thêm và trả về *false*.

Lưu ý: Yêu cầu này chỉ được tính đúng khi sinh viên xét điều kiện và thêm đối tượng Student mới vào danh sách đúng vị trí có thứ tự. Sinh viên **không được phép** thêm đối tượng vào vị trí ngẫu nhiên và sắp xếp lại danh sách.

Sinh viên hiện thực phương thức trên, trong lớp **Helper** đã định nghĩa sẵn phương thức đọc file gọi đến phương thức này. Các lệnh gọi phương thức của lớp **Helper** trong phương thức **main** sẽ cho kết quả ghi ra file “Req1.txt” như file output mẫu.

Lưu ý: Đây là phương thức sinh viên phải định nghĩa được mới bắt đầu tính điểm cho bài, nếu sinh viên định nghĩa sai đồng nghĩa với việc không đọc được file thành danh sách liên kết chứa các đối tượng **Student** thì các yêu cầu bên dưới không được tính điểm.

Các yêu cầu bên dưới thao tác trực tiếp trên các danh sách được thêm sinh viên bằng phương thức này.

2. YÊU CẦU 2 (2 điểm)

Hiện thực phương thức **public boolean deleteStudent(int id)** xóa nút chứa sinh viên theo mã số sinh viên **id** truyền vào. Nếu xóa thành công thì trả về *true*, ngược lại danh sách không chứa sinh viên có mã sinh viên truyền vào thì không xóa và trả về *false*. Trường hợp danh sách rỗng sinh viên ném ra ngoại lệ *NoSuchElementException*.

Sinh viên hiện thực phương thức trên và chạy các lệnh gọi phương thức trong phương thức **main** sau đó so sánh kết quả ghi ra file “Req2.txt” với file trong folder output mẫu. Tuy nhiên các lệnh cho sẵn không đảm bảo đầy đủ trường hợp của phương thức này, sinh viên có thể điều chỉnh các câu lệnh của yêu cầu này trong phương thức **main** để kiểm tra thêm các trường hợp khác.

3. YÊU CẦU 3 (1 điểm)

Hiện thực phương thức **public boolean modifyName(int id, String name)** dùng để điều chỉnh tên của sinh viên có mã số sinh viên **id** theo tham số **name** truyền vào. Trường hợp sửa thành công thì trả về *true*, ngược lại không có sinh viên theo mã số sinh viên truyền vào thì trả về *false*.

Sinh viên hiện thực phương thức trên và chạy các lệnh gọi phương thức trong phương thức **main** sau đó so sánh kết quả ghi ra file “Req3.txt” với file trong folder output mẫu.

4. YÊU CẦU 4 (1 điểm)

Hiện thực phương thức

public Student getHighestScore()

trả về sinh viên đầu tiên có điểm tích lũy cao nhất trong danh sách liên kết. (Danh sách liên kết trong testcase chấm cho yêu cầu này luôn có ít nhất 01 nút)

Sinh viên hiện thực phương thức trên và chạy các lệnh gọi phương thức trong phương thức **main** sau đó so sánh kết quả ghi ra file “Req4.txt” với file trong folder output mẫu.

5. YÊU CẦU 5 (1 điểm)

Hiện thực phương thức

public StudentLinkedList getGraduateStudents(int year)

trả về danh sách liên kết sinh viên chứa các sinh viên đã tốt nghiệp tính tới năm **year** truyền vào. Điều kiện để xét sinh viên đã tốt nghiệp là có năm truyền vào lớn hơn 4 năm so với năm nhập học. Giả sử không có sinh viên nào tốt nghiệp trễ tiến độ. Nếu không có sinh viên nào thỏa điều kiện thì trả về danh sách có **head** là *null*.

Sinh viên hiện thực phương thức trên và chạy các lệnh gọi phương thức trong phương thức **main** sau đó so sánh kết quả ghi ra file “Req5.txt” với file trong folder output mẫu.

6. YÊU CẦU 6 (1 điểm)

Hiện thực phương thức

public StudentLinkedList findByName(String str)

trả về danh sách liên kết sinh viên chứa các sinh viên có tên trùng khớp hoàn toàn với chuỗi **str** truyền vào. Biết tên là từ cuối cùng trong chuỗi họ và tên. Nếu không có sinh viên nào thỏa điều kiện thì trả về danh sách có **head** là *null*.

Sinh viên hiện thực phương thức trên và chạy các lệnh gọi phương thức trong phương thức **main** sau đó so sánh kết quả ghi ra file “Req6.txt” với file trong folder output mẫu.

7. YÊU CẦU 7 (2 điểm)

Lưu ý: Đây là yêu cầu đòi hỏi khi hiện thực phải hiện thực đúng hoàn toàn, nếu sinh viên cố làm yêu cầu này mà chưa hoàn thiện sẽ có khả năng ảnh hưởng dẫn đến sai tất cả các yêu cầu bên trên. Do đó sinh viên chỉ nên thực hiện yêu cầu này cuối cùng và sao lưu lại một bản của các yêu cầu bên trên để tránh trường hợp sinh viên không làm được yêu cầu này sẽ không ảnh hưởng đến các yêu cầu trước đó.

Hiện thực phương thức

public boolean undo()

thực hiện hoàn tác thao tác trước đó vừa thực hiện trên danh sách liên kết. Phương thức này chỉ yêu cầu hoạt động với thao tác thêm nút và thao tác xóa nút. Nếu thực hiện thành công thì trả về *true*, trường hợp không có thao tác nào để hoàn tác thì trả về *false*.

*Gợi ý: Sinh viên sử dụng **Stack<Pair> actions** để lưu vết lại các thao tác thêm nút hoặc xóa nút. Lớp **Pair** có sẵn hai thuộc tính để chứa nút **node** vừa thao tác và chuỗi **action** để lưu lại thao tác là thêm hay xóa.*

Sinh viên hiện thực phương thức trên và chạy các lệnh gọi phương thức trong phương thức **main** sau đó so sánh kết quả ghi ra file “Req7.txt” với file trong folder output mẫu.

VIII. Lưu ý kiểm tra trước khi nộp bài

- Nếu sinh viên không thực hiện được yêu cầu nào thì để nguyên phương thức của yêu cầu đó, **TUYỆT ĐỐI KHÔNG XÓA PHƯƠNG THỨC CỦA YÊU CẦU** sẽ dẫn đến lỗi khi chạy phương thức **main**. Trước khi nộp phải kiểm tra chạy được với phương thức **main** được cho sẵn.
- Tất cả các file ReqX.txt ($X = \{1,2,3,4,5,6,7\}$) được ghi ra cùng cấp thư mục với thư mục chứa các file source code. Đối với sinh viên sử dụng IDE (Eclipse, Netbean, ...) phải đảm bảo file chạy được bằng command prompt, đảm bảo bài làm không nằm trong package, vị trí ghi file kết quả ReqX.txt phải nằm cùng thư mục với file code.
- File kết quả ghi đúng thư mục khi chạy chương trình sẽ có dạng như sau:

Name	Date modified	Type	Size
input	20/07/2022 02:27	File folder	
Helper.class	20/07/2022 05:00	CLASS File	2 KB
Helper.java	20/07/2022 02:12	JAVA File	2 KB
Main.class	20/07/2022 05:00	CLASS File	3 KB
Main.java	20/07/2022 03:15	JAVA File	3 KB
Node.class	20/07/2022 05:00	CLASS File	1 KB
Node.java	12/07/2022 17:51	JAVA File	1 KB
Pair.class	20/07/2022 05:00	CLASS File	2 KB
Pair.java	20/07/2022 03:35	JAVA File	1 KB
Req1.txt	20/07/2022 02:29	Text Document	2 KB
Req2.txt	20/07/2022 02:29	Text Document	1 KB
Req3.txt	20/07/2022 02:29	Text Document	2 KB
Req4.txt	20/07/2022 02:29	Text Document	1 KB
Req5.txt	20/07/2022 02:29	Text Document	1 KB
Req6.txt	20/07/2022 02:29	Text Document	1 KB
Req7.txt	20/07/2022 02:29	Text Document	1 KB
Student.class	20/07/2022 05:00	CLASS File	2 KB
Student.java	12/07/2022 17:51	JAVA File	2 KB
StudentLinkedList.class	20/07/2022 05:00	CLASS File	4 KB
StudentLinkedList.java	20/07/2022 02:04	JAVA File	6 KB


IX. Hướng dẫn nộp bài

- Khi nộp bài sinh viên nộp lại file *StudentLinkedList.java*, **không nộp kèm bất cứ file nào khác và tuyệt đối không được sửa tên file này.**
- **Sinh viên đặt file bài làm vào thư mục MSSV_HoTen** (HoTen viết liền, không dấu) và nén lại với định dạng **.zip** nộp theo sự hướng dẫn của giảng viên thực hành.
- Trường hợp làm sai yêu cầu nộp bài (đặt tên thư mục sai, không để bài làm vào thư mục khi nộp, nộp dư file, ...) thì bài làm của sinh viên sẽ bị **0 điểm**.
- File nộp đúng sẽ như sau:

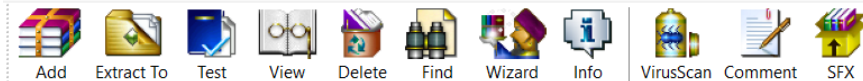
- File nén nộp bài:


 51403039_DungCamQuang.zip 20/07/2022 05:06 WinRAR ZIP archive 2 KB

- Bên trong file nén:

 51403039_DungCamQuang.zip (evaluation copy)


File Commands Tools Favorites Options Help



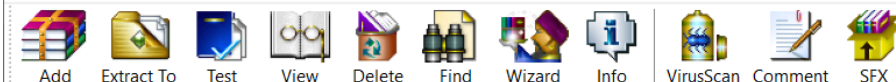
 51403039_DungCamQuang.zip - ZIP archive, unpacked size 5,313 bytes


Name	Size	Packed	Type	Modified	CRC32
..			Local Disk		
51403039_DungCamQuang			File folder	20/07/2022 05:02	

- Bên trong thư mục:

 51403039_DungCamQuang.zip (evaluation copy)

File Commands Tools Favorites Options Help



 51403039_DungCamQuang.zip\51403039_DungCamQuang - ZIP archive, unpacked size 5,313 bytes

Name	Size	Packed	Type	Modified	CRC32
..			Local Disk		
StudentLinkedList.java	5,313	983	JAVA File	20/07/2022 02:04	F82BDC78

X. Đánh giá và quy định

- Bài làm sẽ được chấm tự động thông qua testcase (file input và output có định dạng như mẫu đã gửi kèm) do đó sinh viên tự chịu trách nhiệm nếu không thực hiện đúng theo hướng dẫn trong đề, đặc biệt là Hướng dẫn nộp bài hoặc tự ý sửa tên các phương thức đã có sẵn dẫn đến bài làm không biên dịch được khi chấm.
- Testcase sử dụng để chấm bài là file khác với file sinh viên đã nhận, sinh viên chỉ được điểm mỗi YÊU CẦU khi chạy ra đúng hoàn toàn kết quả của yêu cầu đó.
- Nếu bài làm của sinh viên biên dịch bị lỗi trên máy chấm thì **0 điểm cả bài**.
- **Tất cả code sẽ được kiểm tra đạo văn. Mọi hành vi sao chép code trên mạng, chép bài bạn hoặc cho bạn chép bài nếu bị phát hiện đều sẽ bị điểm 0 vào điểm Quá trình 2 hoặc cấm thi cuối kì.**
- Nếu bài làm của sinh viên có dấu hiệu sao chép trên mạng hoặc sao chép nhau, sinh viên sẽ được gọi lên phỏng vấn code để chứng minh bài làm là của mình.
- **Hạn chót nộp bài: 23h00 ngày 04/08/2022.**

-- HẾT --