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 2

CUỘC KHÁNG CHIẾN CHỐNG QUÂN MÔNG NGUYÊN - LẦN THỨ BA

TP. HỒ CHÍ MINH, THÁNG 05/2021

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:

- Lập trình hướng đối tượng
- Danh sách liên kết
- Quản lý mảng động

2 Dẫn nhập

Hai lần xâm chiếm Đại Việt đều thất bại, hoàng đế nhà Nguyên càng tức giận nên quyết tâm đánh Đại Việt lần thứ ba để trả thù. Hốt Tất Liệt đình chỉ tiến công Nhật Bản, tập trung lực lượng tiến công Đại Việt. Cuộc chiến tranh diễn ra trên lãnh thổ Đại Việt từ tháng 12 năm 1287 đến cuối tháng 4 năm 1288.

Trong cuộc chiến lần này, nhà Nguyên huy động hơn 30 vạn quân và nhiều danh tướng do Thoát Hoan làm tổng chỉ huy. Theo Nguyên sử, ngoài việc huy động lại những quân lính trong lần chinh phạt thứ hai thoát được về Trung Quốc, nhà Nguyên còn huy động thêm quân từ Mông Cổ và Hán, quân nhà Nam Tống cũ đầu hàng theo Nguyên, quân của Vân Nam, ... Về thủy quân, quân Nguyên cũng huy động nhiều thuyền vận tải chở thạch lương do Trương Văn Hổ chỉ huy đi cùng Ô Mã Nhi. Hốt Tất Liệt cũng đã căn dặn Thoát Hoan không được cho Giao Chỉ là một nước nhỏ mà khinh thường.

Đứng trước nguy cơ bị xâm lược, vua Trần khẩn trương chuẩn bị đánh giặc. Phía Đại Việt đã tiến hành tổng động viên. Tướng chỉ huy toàn bộ quân đội là Hưng Đạo Vương Trần Quốc Tuấn. Trần Nhân Tông đại xá thiên hạ, chỉ trừ những người từng hàng quân Nguyên đều không được đại xá. Nhiều tù nhân được ra tình nguyện tòng quân ra mặt trận để báo ơn. Trần Quốc Tuấn với những kinh nghiệm tác chiến thu được sau khi đánh bại quân Nguyên 2 năm trước, sau khi phân tích tình hình quân Nguyên, đã tự tin tâu với vua Trần: *"Thế giặc năm nay dễ phá"*.

Cuối tháng 12/1287, khoảng 30 vạn quân Nguyên tiến vào nước ta. Cánh quân bộ do Thoát Hoan chỉ huy, vượt biên giới đánh vào Lạng Sơn, Bắc Giang rồi kéo về Vạn Kiếp. Cánh quân

thủy do Ô Mã Nhi chỉ huy theo đường biển tiến vào sông Bạch Đằng, rồi tiến về Vạn Kiếp. Tại Vân Đồn, Trần Khánh Dư chỉ huy quân mai phục, khi đoàn thuyền lương của Trương Văn Hổ đến, quân Trần đánh dữ dội. Phần lớn thuyền lương của giặc bị đánh đắm, số còn lại bị quân ta chiếm. Cuối tháng 1/1288, Thoát Hoan kéo quân vào kinh thành Thăng Long trống vắng. Sau trận Vân Đồn, tình thế quân Nguyên ngày càng khó khăn, nhiều nơi xung yếu bị quân Trần tấn công chiếm lại, lương thực ngày càng cạn kiệt, quân giặc ở Thăng Long đứng trước tình thế bị cô lập. Thoát Hoan quyết định rút quân về Vạn Kiếp và từ đây rút quân về nước theo hai đường thuỷ, bộ

Nhà Trần mở cuộc phản công ở cả hai mặt trận thuỷ, bộ. Quân ta bố trí, mai phục sẵn ở sông Bạch Đằng. Tháng 4/1288, đoàn thuyền của Ô Mã Nhi lọt vào trận địa bãi cọc trên sông Bạch Đằng do quân Trần bố trí từ trước. Cuộc chiến đấu ác liệt diễn ra, Ô Mã Nhi bị bắt sống. Trên bộ, Thoát Hoan dẫn quân từ Vạn Kiếp theo hướng Lạng Sơn rút về Trung Quốc, bị quân dân ta liên tục chặn đánh.

Quân Nguyên thất bại thảm hại, bị đập tan mộng xâm lược Đại Việt, ba lần kháng chiến chống quân xâm lược Mông Nguyên kết thúc.

3 Dữ liệu đầu vào

3.1 Mô ta các file trong bài làm

Trong bài tập lớn lần này, sinh viên sẽ phải hoàn thành 5 nhiệm vụ, từ nhiệm vụ 0 đến nhiệm vụ 4.

- **main.cpp**: là file chứa hàm main, nơi bắt đầu chạy chương trình. Hàm main sẽ gọi lần lượt các hàm để giải quyết các nhiệm vụ, các hàm này sẽ lần lượt nằm trong các file header tương ứng. Ví dụ, hàm *solveTask0* sẽ nằm trong file *task0.h*.
- **dataStructure.h**: là file chứa định nghĩa các struct được sử dụng trong bài tập lớn lần này. Trong đó:
  - struct Soldier: chứa thông tin cho một binh lính, gồm HP (máu), isSpecial (lính đặc biệt, true nếu phải và false nếu ngược lại), ID (ID của lính đó).
  - struct SoldierNode: chứa thông tin của node để sử dụng trong danh sách liên kết đơn **SLinkedList**.
  - struct SLinkedList: chứa thông tin của danh sách liên kết đơn, mà tại đó mỗi node lưu trữ thông tin về một binh lính.

  - struct Array: chứa thông tin của mảng động, mà tại đó, mỗi phần tử trong mảng lưu trữ thông tin của một binh lính.

- **SLLDataController.h** và **ArrayDataController.h**: là những file chứa các hàm để thao tác lần lượt trên danh sách liên kết đơn và mảng.
- **thirdBattle.h**: là những file chứa các hàm cho nhiệm vụ từ 1 đến 4.
- **task<i>.h**: trong đó, i chạy từ 0 đến 4, là file chứa hàm để xử lý việc đọc testcase và chạy hàm liên quan đến nhiệm vụ, xuất ra kết quả.
- **testcase<i>.txt**: trong đó, i chạy từ 0 đến 4, là những file chứa testcase. Lưu ý, trong những file tesecase này, cuối mỗi dòng KHÔNG tồn tại khoảng trắng nào.

Trong bài tập lớn lần này, sinh viên phải hoàn thành các hàm nằm trong các file **SLLDataController.h**, **ArrayDataController.h**, **thirdBattle.h**, và cũng chỉ cần nộp những file này. Ngoài ra, sinh viên không được phép thêm thư viện nào, tuy nhiên, được viết thêm hàm vào ba file nêu trên, chỗ viết thêm hàm được chú thích rõ trong từng file.

Việc giải thích dữ liệu nhập cho từng testcase sẽ được nêu rõ theo từng nhiệm vụ.

4 Nhiệm vụ

Sinh viên được yêu cầu xây dựng một chương trình giả tưởng trên ngôn ngữ C++ để mô phỏng lại quá trình chuẩn bị binh lính, thông qua các nhiệm vụ được mô tả bên dưới. Lưu ý, một nhiệm vụ có thể có nhiều hàm cần hiện thực.

4.1 Nhiệm vụ 0

Trong nhiệm vụ này, sinh viên cần phải hoàn thành các hàm nằm trong hai file **SLLDataController.h** và **ArrayDataController.h** để có được những công cụ quản lý danh sách liên kết và mảng động cần thiết cho các nhiệm vụ sau.

Mô tả hàm trong file **SLLDataController.h**, lưu ý, phần tử cuối cùng của danh sách liên kết phải trỏ về NULL:

- **print**:
  - Tham số: danh sách liên kết *list*
  - Hàm sẽ in ra màn hình các phần tử nằm trong danh sách liên kết được truyền vào
  - Ký hiệu trong testcase: PRINT

- **insertAt**:
  - Tham số: danh sách liên kết *list*, thông tin về binh lính *element*, vị trí cần thêm *pos*
  - Hàm sẽ thêm vào *list* một binh lính đúng vị trí cho sẵn, sau khi thêm, binh lính sẽ nằm ở đúng vị trí *pos*. Hàm trả về 1 nếu thêm thành công và 0 nếu ngược lại.
  - Ký hiệu trong testcase: INSERTAT[space]<pos>[space]<HP>[space]<isSpecial>[space]<ID>
- **removeAt**:
  - Tham số: danh sách liên kết *list*, vị trí cần xoá *pos*
  - Hàm sẽ xoá khỏi *list* một binh lính tại vị trí *pos*. Hàm trả về 1 nếu xoá thành công và 0 nếu ngược lại.
  - Ký hiệu trong testcase: REMOVEAT[space]<pos>
- **removeFirstItemWithHP**:
  - Tham số: danh sách liên kết *list*, HP binh lính cần xoá *HP*
  - Hàm sẽ xoá khỏi *list* binh lính đầu tiên tìm được có giá trị HP bằng với *HP* của tham số. Hàm trả về 1 nếu xoá thành công và 0 nếu ngược lại.
  - Ký hiệu trong testcase: REMOVEHP[space]<HP>
- **indexOf**:
  - Tham số: danh sách liên kết *list*, thông tin binh lính cần tìm *soldier*
  - Hàm sẽ tìm trong *list* vị trí binh lính đầu tiên tìm được có ba thông tin ID, HP và isSpecial trùng với ba thông tin được lưu giữ trong *soldier* của tham số truyền vào. Hàm trả về -1 nếu không tìm được binh lính.
  - Ký hiệu trong testcase: INDEX[space]<HP>[space]<isSpecial>[space]<ID>
- **size**:
  - Tham số: danh sách liên kết *list*
  - Hàm trả về số lượng binh lính có trong danh sách liên kết
  - Ký hiệu trong testcase: SIZE
- **empty**:
  - Tham số: danh sách liên kết *list*
  - Hàm trả về true nếu danh sách liên kết rỗng và false nếu ngược lại
  - Ký hiệu trong testcase: EMPTY
- **clear**:
  - Tham số: danh sách liên kết *list*
  - Hàm sẽ xoá hết các phần tử có trong danh sách

  - Ký hiệu trong testcase: CLEAR

- **getIDAt**:
  - Tham số: danh sách liên kết *list*, vị trí *pos*
  - Trả về ID của binh lính tại vị trí *pos*, trả về -1 nếu không tìm thấy vị trí trong danh sách liên kết.
  - Ký hiệu trong testcase: GETID[space]<pos>
- **getHPAt**:
  - Tham số: danh sách liên kết *list*, vị trí *pos*
  - Trả về HP của binh lính tại vị trí *pos*, trả về -1 nếu không tìm thấy vị trí trong danh sách liên kết.
  - Ký hiệu trong testcase: GETHP[space]<pos>
- **setHPAt**:
  - Tham số: danh sách liên kết *list*, máu *HP*, vị trí *pos*
  - Gán máu của binh lính tại vị trí *pos* bằng *HP*. Trả về 1 nếu gán thành công và 0 nếu ngược lại
  - Ký hiệu trong testcase: SETHP[space]<pos>[space]<HP>
- **contains**:
  - Tham số: danh sách liên kết *list*, thông tin binh lính cần tìm *soldier*
  - Hàm trả về 1 nếu có thể tìm thấy binh lính *soldier* trong danh sách liên kết và 0 nếu ngược lại.
  - Ký hiệu trong testcase: CONTAINS[space]<HP>[space]<isSpecial>[space]<ID>

Mô tả hàm trong file **ArrayDataController.h**:

- **print**:
  - Tham số: mảng *array*
  - Hàm sẽ in ra màn hình các phần tử nằm trong mảng được truyền vào
  - Ký hiệu trong testcase: PRINT
- **initArray**:
  - Tham số: mảng *array*, sức chứa tối đa *cap*
  - Hàm sẽ khởi tạo mảng động với số lượng phần tử *cap*
  - Ký hiệu trong testcase: INIT[space]<cap>
- **insertAt**:

  - Tham số: mảng *array*, thông tin về binh lính *element*, vị trí cần thêm *pos*
  - Hàm sẽ thêm vào *array* một binh lính đúng vị trí cho sẵn, sau khi thêm, binh lính sẽ nằm ở đúng vị trí *pos*. Hàm trả về 1 nếu thêm thành công và 0 nếu ngược lại.
  - Ký hiệu trong testcase: INSERTAT[space]<pos>[space]<HP>[space]<isSpecial>[space]<ID>

- **removeAt**:
  - Tham số: mảng *array*, vị trí cần xoá *pos*
  - Hàm sẽ xoá khỏi *array* một binh lính tại vị trí *pos*. Hàm trả về 1 nếu xoá thành công và 0 nếu ngược lại.
  - Ký hiệu trong testcase: REMOVEAT[space]<pos>

- **removeFirstItemWithHP**:
  - Tham số: mảng *array*, HP binh lính cần xoá *HP*
  - Hàm sẽ xoá khỏi *array* binh lính đầu tiên tìm được có giá trị HP bằng với *HP* của tham số. Hàm trả về 1 nếu xoá thành công và 0 nếu ngược lại.
  - Ký hiệu trong testcase: REMOVEHP[space]<HP>

- **indexOf**:
  - Tham số: mảng *array*, thông tin binh lính cần tìm *soldier*
  - Hàm sẽ tìm trong *array* vị trí binh lính đầu tiên tìm được có ba thông tin ID, HP và isSpecial trùng với ba thông tin được lưu giữ trong *soldier* của tham số truyền vào. Hàm trả về -1 nếu không tìm được binh lính.
  - Ký hiệu trong testcase: INDEX[space]<HP>[space]<isSpecial>[space]<ID>

- **size**:
  - Tham số: mảng *array*
  - Hàm trả về số lượng binh lính có trong mảng
  - Ký hiệu trong testcase: SIZE

- **empty**:
  - Tham số: mảng *array*
  - Hàm trả về true nếu mảng rỗng và false nếu ngược lại
  - Ký hiệu trong testcase: EMPTY

- **clear**:
  - Tham số: mảng *array*
  - Hàm sẽ xoá hết các phần tử có trong mảng, đồng thời biến con trỏ đang trỏ tới mảng trong array sẽ phải trỏ về NULL.
  - Ký hiệu trong testcase: CLEAR

- **getIDAt**:
  - Tham số: mảng *array*, vị trí *pos*
  - Trả về ID của binh lính tại vị trí *pos*, trả về -1 nếu không tìm thấy vị trí trong mảng.
  - Ký hiệu trong testcase: GETID[space]<pos>
- **getHPAt**:
  - Tham số: mảng *array*, vị trí *pos*
  - Trả về HP của binh lính tại vị trí *pos*, trả về -1 nếu không tìm thấy vị trí trong mảng.
  - Ký hiệu trong testcase: GETHP[space]<pos>
- **setHPAt**:
  - Tham số: mảng *array*, máu *HP*, vị trí *pos*
  - Gán máu của binh lính tại vị trí *pos* bằng *HP*. Trả về 1 nếu gán thành công và 0 nếu ngược lại
  - Ký hiệu trong testcase: SETHP[space]<pos>[space]<HP>
- **contains**:
  - Tham số: mảng *array*, thông tin binh lính cần tìm *soldier*
  - Hàm trả về 1 nếu có thể tìm thấy binh lính *soldier* trong mảng và 0 nếu ngược lại.
  - Ký hiệu trong testcase: CONTAINS[space]<HP>[space]<isSpecial>[space]<ID>

Trong nhiệm vụ này, testcase có 1+N dòng, với dòng đầu tiên chứa số lượng câu lệnh (N), và N dòng tiếp tiếp theo là các câu lệnh để thao tác. Lưu ý, ở đầu mỗi câu lệnh sẽ có các ký tự để chỉ rõ câu lệnh được thực hiện trên cấu trúc nào, "A" với array và "L" với list.

Ví dụ 1:

```
3
A INIT 10
L PRINT
A INSERTAT 3 2 1 H
```

Testcase có 3 dòng lệnh, dòng đầu sẽ khởi tạo array với capacity là 10, dòng thứ hai sẽ in list ra, và dòng thứ 3 sẽ thêm một binh lính vào mảng tại vị trí số 3, với HP, isSpecial và ID lần lượt là 2, 1 và H.

4.2 Nhiệm vụ 1

Sinh viên có thể tìm thấy các hàm liên quan đến nhiệm vụ này trong file **thirdBattle.h**. Trong nhiệm vụ này, ta tiến hành xây dựng một cấu trúc dữ liệu để quản lý binh lính từ mảng.

Mảng này sẽ có tính chất Last In First Out (LIFO) - nghĩa là phần tử cuối cùng được thêm vào mảng sẽ là phần tử đầu tiên rời khỏi mảng. Trong nhiệm vụ này, sinh viên cần hiện thực ba hàm:

- **push**:
  - Tham số: mảng *array*, thông tin binh lính *soldier*
  - Hàm sẽ thêm một binh lính vào mảng, trả về 1 nếu thêm thành công và 0 nếu ngược lại
  - Ký hiệu trong testcase: PUSH[space]<HP>[space]<isSpecial>[space]<ID>
- **pop**:
  - Tham số: mảng *array*
  - Xoá một binh lính khỏi mảng, trả về 1 nếu xoá thành công và 0 nếu ngược lại
  - Ký hiệu trong testcase: POP
- **top**:
  - Tham số: mảng *array*
  - Hàm trả về binh lính có thời gian ở trong mảng ngắn nhất, trả về mặc định đã có sẵn trong hàm nếu không thể thực hiện thao tác.
  - Ký hiệu trong testcase: TOP

Lưu ý, hàm **push** và **pop** phải theo tiêu chí LIFO được đề cập ở trên. Trong nhiệm vụ này, testcase có 1+N dòng, với dòng đầu tiên chứa số lượng câu lệnh (N), và N dòng tiếp tiếp theo là các câu lệnh để thao tác.

Ví dụ 2:

```
3
PUSH 10 0 H
TOP
POP
```

Testcase có 3 dòng lệnh, dòng đầu tiên sẽ thêm binh lính có chỉ số ID="H", HP=10 và isSpecial=0 vào mảng, dòng thứ hai sẽ lấy thông tin binh lính ra khỏi mảng, và dòng thứ

ba sẽ xoá đi một binh lính.

4.3 Nhiệm vụ 2

Sinh viên có thể tìm thấy các hàm liên quan đến nhiệm vụ này trong file **thirdBattle.h**. Trong nhiệm vụ này, ta tiến hành xây dựng một cấu trúc dữ liệu để quản lý binh lính từ danh sách liên kết.

Danh sách liên kết này có tính chất First In First Out (FIFO) - nghĩa là phần tử đầu tiên được thêm vào danh sách sẽ là phần tử đầu tiên rời khỏi danh sách. Trong nhiệm vụ này, sinh viên cần hiện thực ba hàm:

- **enqueue**:
  - Tham số: danh sách liên kết *list*, thông tin binh lính *soldier*
  - Hàm sẽ thêm một binh lính vào danh sách liên kết, trả về 1 nếu thêm thành công và 0 nếu ngược lại
  - Ký hiệu trong testcase: EN[space]<HP>[space]<isSpecial>[space]<ID>
- **dequeue**:
  - Tham số: danh sách liên kết *list*
  - Xoá một binh lính khỏi danh sách, trả về 1 nếu xoá thành công và 0 nếu ngược lại
  - Ký hiệu trong testcase: DE
- **front**:
  - Tham số: danh sách liên kết *list*
  - Hàm trả về binh lính có thời gian ở trong danh sách lâu nhất, trả về mặc định đã có sẵn trong hàm nếu không thể thực hiện thao tác.
  - Ký hiệu trong testcase: FRONT

Lưu ý, hàm **enqueue** và **dequeue** phải theo tiêu chí FIFO được đề cập ở trên. Trong nhiệm vụ này, testcase có 1+N dòng, với dòng đầu tiên chứa số lượng câu lệnh (N), và N dòng tiếp tiếp theo là các câu lệnh để thao tác.

Ví dụ 3:

```
3
EN 10 0 H
FRONT
```

DE

Testcase có 3 dòng lệnh, dòng đầu tiên sẽ thêm binh lính có chỉ số ID="H", HP=10 và isSpecial=0 vào danh sách, dòng thứ hai sẽ lấy thông tin binh lính ra khỏi danh sách, và dòng thứ ba sẽ xoá đi một binh lính.

4.4 Nhiệm vụ 3

Sinh viên có thể tìm thấy các hàm liên quan đến nhiệm vụ này trong file **thirdBattle.h**. Trong nhiệm vụ này, ta tiến hành đảo ngược dữ liệu trong một danh sách liên kết thông qua hàm **reverse**.

Testcase có N dòng chứa thông tin của N binh lính để thêm vào danh sách liên kết. Việc thêm vào danh sách đã được thực hiện trong file **task3.h**. Sinh viên tiến hành hiện thực hàm **reverse** để đảo ngược danh sách này.

Ví dụ 4:
3
12 1 K
23 0 F
2 1 G

Testcase chứa 3 binh lính với các thông tin (HP, isSpecial, ID) theo thứ tự (12, 1, K), (23, 0, F), (2, 1, G).
Khi ba binh lính này được đọc vào, *list* được lưu trữ như sau (nếu sinh viên hiện thực đúng hàm **insertAt** trong nhiệm vụ 0): $(2, 1, G) \rightarrow (23, 0, F) \rightarrow (12, 1, K) \rightarrow$ NULL

Hàm **reverse** sẽ tiến hành đảo ngược danh sách trên để trở thành: NULL $\leftarrow (2, 1, G) \leftarrow (23, 0, F) \leftarrow (12, 1, K)$

4.5 Nhiệm vụ 4

Sinh viên có thể tìm thấy các hàm liên quan đến nhiệm vụ này trong file **thirdBattle.h**. Trong nhiệm vụ này, ta tiến hàm hợp hai danh sách binh lính đã có thứ tự lại thành một danh sách

mới có thứ tự.

Các binh linh được sắp xếp theo thứ tự ưu tiên như sau trong danh sách:

- Ưu tiên 1: Giữa hai binh lính, người nào có HP thấp hơn sẽ ở phía trước (index nhỏ hơn).
- Ưu tiên 2: Nếu hai binh lính trùng HP, binh lính nào có cờ isSpecial là false sẽ đứng trước (index nhỏ hơn).
- Ưu tiên 3: Nếu hai binh lính trùng HP, trùng isSpecial, thì binh lính nào có ID nhỏ hơn theo bảng chữ cái sẽ đứng trước (ID chỉ có 1 ký tự).

Testcase chứa thông tin của các binh lính và được thêm vào *list1* và *list2*, có 1+N+1+M dòng, tương ứng với N binh lính của *list1* và M binh lính của *list2*. Việc thêm vào danh sách đã được thực hiện trong file **task4.h**. Sinh viên tiến hành hiện thực hàm **merge** để hợp hai danh sách có thứ tự này thành một danh sách mới có thứ tự nêu trên đề miêu tả. Lưu ý, hai danh sách con ban đầu cũng tuân thủ theo thứ tự ở trên.

Ví dụ 5:

```
3
2 0 A
2 0 B
2 1 A
2
1 0 D
2 1 E
```

Testcase chứa 5 binh lính với các thông tin (HP, isSpecial, ID)
list1 được lưu trữ như sau (nếu sinh viên hiện thực đúng hàm **insertAt** trong nhiệm vụ 0): $(2, 0, A) \rightarrow (2, 0, B) \rightarrow (2, 1, A) \rightarrow$ NULL
list2 được lưu trữ như sau (nếu sinh viên hiện thực đúng hàm **insertAt** trong nhiệm vụ 0): $(1, 0, D) \rightarrow (2, 1, E) \rightarrow$ NULL

Sau khi hợp nhất, danh sách mới có dạng: $(1, 0, D) \rightarrow (2, 0, A) \rightarrow (2, 0, B) \rightarrow (2, 1, A) \rightarrow (2, 1, E) \rightarrow$ NULL

5 Nộp bài

Sinh viên nộp 3 tập tin: **SLLDataController.h**, **ArrayDataController.h**, và **thirdBattle.h** trên site `Kỹ thuật lập trình (CO1027)_HK202` trên hệ thống BKeL.

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**.

6 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.

7 Lưu ý về bài tập lớn

- Sinh viên có thể thay đổi thứ tự hàm, viết hàm phụ, hoặc thêm prototype ở đầu file nộp, nhưng không được đổi tên các hàm đã có sẵn.
- Capacity ý chỉ sức chứa tối đa, còn size chỉ số lượng phần tử của mảng tại một thời điểm.

- Việc insert vào mảng hay danh sách liên kết, sinh viên phải đảm bảo rằng các phần tử liên tục nhau (không có NULL giữa hai phần tử bất kỳ trong danh sách liên kết, hoặc không có phần tử rác giữa hai phần tử bất kỳ trong mảng).
- Để tránh những lỗi đáng tiếc có thể xảy ra khi chạy code, sinh viên vui lòng tăng capacity theo quy luật: capacity mới = capacity cũ * 1.5, và không nên khởi tạo mảng với số phần tử quá lớn.

8 Thay đổi so với phiên bản trước

Tài liệu

[1] Lê Văn Siêu (2004), Việt Nam văn minh sử cương, Nhà xuất bản Thanh niên.

[2] Hà Văn Tấn và Phạm Thị Tâm (1972), Cuộc kháng chiến chống xâm lược Nguyên Mông thế kỷ XIII, Nhà xuất bản Quân đội Nhân dân, bản in lại năm 2003.

——————————HẾT——————————