Đồ án thực hành (Phòng C201)

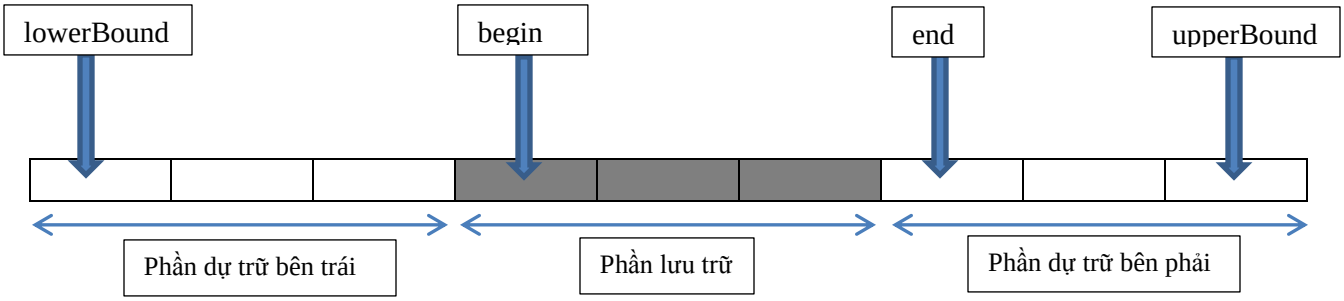
Đề bài: (SV phải đọc kỹ phần quy định cuối file)

1. CUSTOM ARRAY

CustomArray là cấu trúc dùng để tối ưu một số thao tác trên mảng động. Một **CustomArray** kích thước N là một mảng động có tổng kích thước là 3N được chia làm 3 phần:

- Phần dự trữ bên trái có N phần tử
- Phần lưu trữ có N phần tử
- Phần dự trữ bên phải có N phần tử (phần tử cuối của phần này thường không được dùng, chỉ để đánh dấu).

Ví dụ: CustomArray có kích thước N = 3



Câu 1: Khai báo lớp **CustomArray** theo cấu trúc mô tả bên trên bao gồm các thuộc tính:

- Kích thước N của phần lưu trữ
- Con trỏ *lowerBound* trỏ đến phần tử đầu tiên của phần dự trữ bên trái
- Con trỏ *begin* trỏ đến phần tử đầu tiên của phần lưu trữ
- Con trỏ *end* trỏ đến phần tử đầu tiên của phần dự trữ bên phải
- Con trỏ *upperBound* trỏ đến phần tử cuối cùng của phần dự trữ bên phải

Lưu ý: Lớp này có thể sử dụng với bất kỳ kiểu dữ liệu nào.

Câu 2: Cài đặt các phương thức khởi tạo và phương thức hủy cho lớp **CustomArray**. Các phương thức khởi tạo gồm:

- Phương thức khởi tạo mặc định.
- Phương thức khởi tạo CustomArray với kích thước N.
- Phương thức khởi tạo sao chép

- Phương thức khởi tạo CustomArray với kích thước N và mỗi phần tử có giá trị mặc định *val*.

Câu 3: Cài đặt các toán tử gồm:

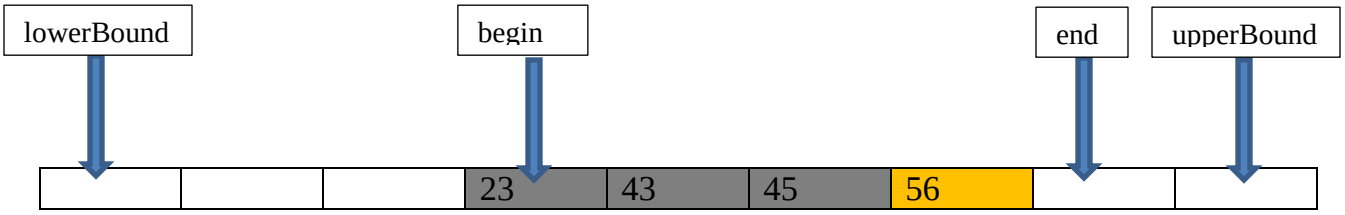
- Toán tử gán (**`operator =`**) sử dụng deep copy.
- Toán tử truy xuất đến phần tử ở vị trí thứ *i* (**`operator []`**)

Câu 4: Cài đặt các phương thức:

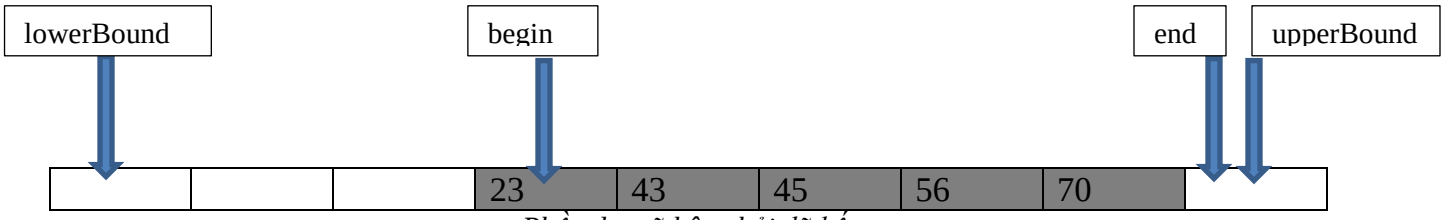
- Phương thức **`size()`** lấy kích thước của CustomArray.
- Phương thức **`begin()`** lấy con trỏ *begin* của CustomArray.
- Phương thức **`end()`** lấy con trỏ *end* của CustomArray.
- Phương thức **`clear()`** xóa tất cả phần tử trong CustomArray, đưa CustomArray về rỗng.
- Phương thức **`empty()`** kiểm tra xem một CustomArray có rỗng hay không.
- Phương thức **`resize()`** điều chỉnh lại kích thước của CustomArray.

Câu 5: Cài đặt phương thức **`push_back()`** dùng để thêm 1 phần tử vào cuối phần lưu trữ của CustomArray. Cách thức thêm như sau:

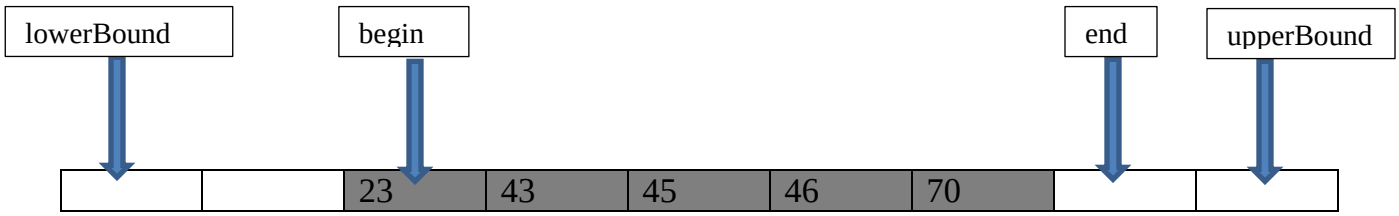
- Nếu phần dự trữ bên phải còn trống (con trỏ end chưa trùng con trỏ upperBound) thì dời con trỏ end sang phải 1 phần tử và thêm phần tử mới vào vị trí cuối phần lưu trữ.



- Nếu phần dự trữ bên phải đã hết (con trỏ end trùng con trỏ upperBound) và phần dự trữ bên trái còn trống (con trỏ begin chưa trùng con trỏ lowerBound) thì tiến hành **cân chỉnh** (adjust) lại CustomArray sao cho phần lưu trữ nằm chính giữa mảng sau đó thêm phần tử như bước trên.

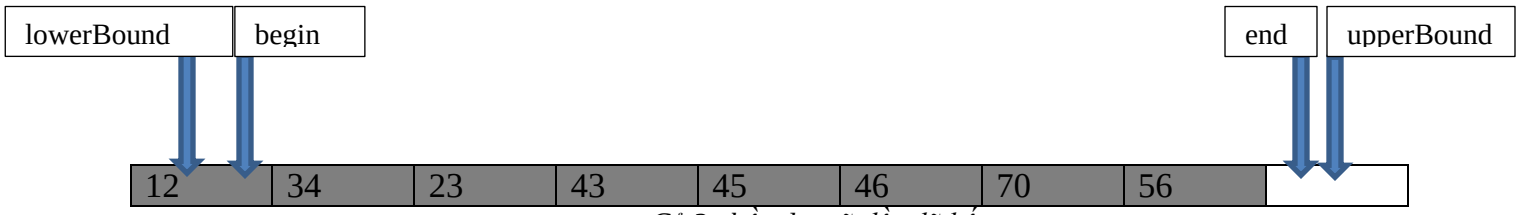


Phần dự trữ bên phải đã hết



CustomArray sau khi cân chỉnh

- Nếu phần dự trữ bên phải đã hết (con trỏ end trùng con trỏ upperBound) và phần dự trữ bên trái cũng đã hết (con trỏ begin trùng con trỏ lowerBound) thì tiến hành cấp phát vùng nhớ với kích thước mới, sao chép dữ liệu từ vùng nhớ cũ, thêm phần tử và hủy vùng nhớ cũ.



Cả 2 phần dự trữ đều đã hết

Câu 6: Cài đặt phương thức **`push_front()`** dùng để thêm 1 phần tử vào đầu phần lưu trữ của CustomArray. Cách thức thêm như sau:

- Nếu phần dự trữ bên trái còn trống (con trỏ begin chưa trùng con trỏ lowerBound) thì dời con trỏ begin sang trái 1 phần tử và thêm phần tử mới vào vị trí đầu phần lưu trữ.
- Nếu phần dự trữ bên trái đã hết (con trỏ begin trùng con trỏ lowerBound) và phần dự trữ bên phải còn trống (con trỏ end chưa trùng con trỏ upperBound) thì tiến hành cân chỉnh lại CustomArray sao cho phần lưu trữ nằm chính giữa mảng, sau đó thêm phần tử như bước trên.
- Nếu phần dự trữ bên trái đã hết (con trỏ begin trùng con trỏ lowerBound) và phần dự trữ bên phải cũng đã hết (con trỏ end trùng con trỏ upperBound) thì tiến hành

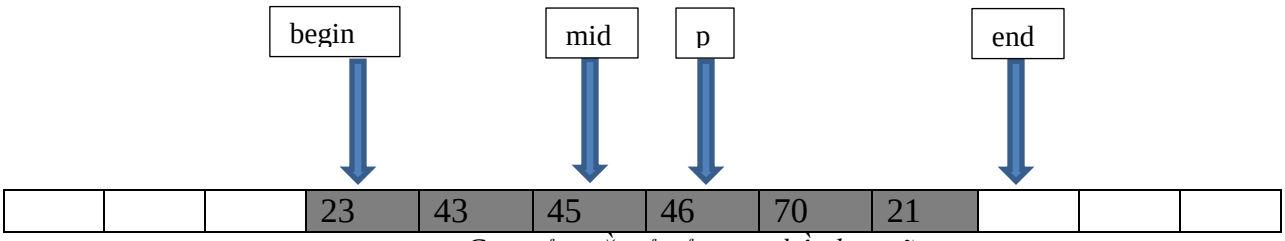
cấp phát vùng nhớ với kích thước mới, sao chép dữ liệu từ vùng nhớ cũ, thêm phần tử và hủy vùng nhớ cũ.

Câu 7: Cài đặt phương thức **`pop_back()`** trả về phần tử cuối của phần lưu trữ đồng thơi xóa phần tử này ra khỏi CustomArray bằng cách dời con trỏ end qua trái 1 phần tử.

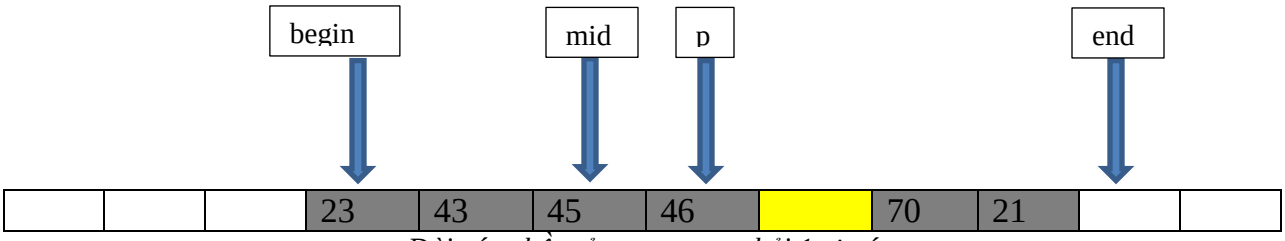
Câu 8: Cài đặt phương thức **`pop_front()`** trả về phần tử đầu của phần lưu trữ đồng thời xóa phần tử này ra khỏi CustomArray bằng cách dời con trỏ begin qua phải 1 phần tử.

Câu 9: Cài đặt phương thức **`insert()`** dùng để thêm 1 phần tử mới vào sau phần tử mà con trỏ p đang trỏ tới. Cách thức thêm như sau:

- Nếu phần dự trữ bên phải vẫn còn và con trỏ p nằm ở nửa sau của phần lưu trữ thì dời các phần tử ở phía sau p sang phải 1 vị trí và thêm phần tử mới vào sau p.

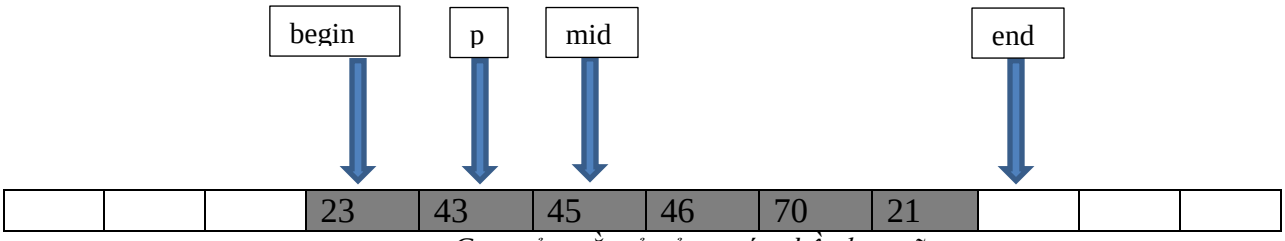


Con trỏ p nằm ở nửa sau phần lưu trữ

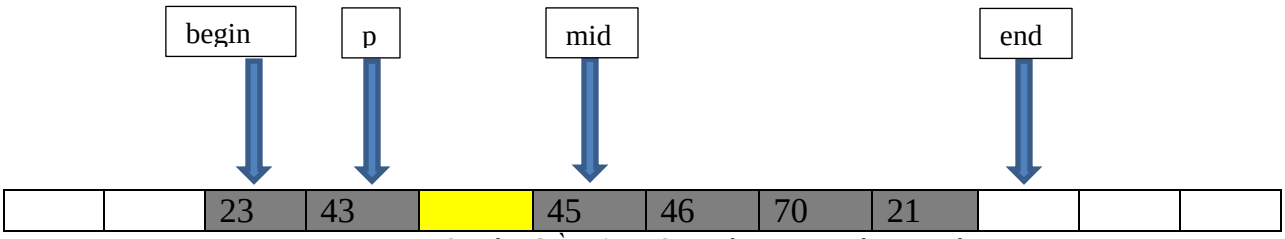


Dời các phần tử sau p sang phải 1 vị trí

- Nếu phần dự trữ bên phải đã hết và con trỏ p nằm ở nửa sau của phần lưu trữ thì tiến hành cân chỉnh CustomArray sau đó thêm giống như trên
- Nếu phần dự trữ bên trái vẫn còn và con trỏ p nằm ở nửa trước của phần lưu trữ thì dời các phần tử ở phía trước p và p sang trái 1 vị trí và thêm phần tử mới vào vị trí cũ của p.



Con trỏ p nằm ở nửa trước phần lưu trữ



Dời các phần tử trước p và p sang trái 1 vị trí

- Nếu phần dự trữ bên trái đã hết và con trỏ p nằm ở nửa trước của phần lưu trữ thì tiến hành cân chỉnh CustomArray sau đó thêm giống như trên.
- Nếu cả 2 phần dự trữ đều đã hết thì cấp phát vùng nhớ với kích thước mới, thêm phần tử theo quy tắc trên và hủy vùng nhớ cũ.

Câu 10: Cài đặt phương thức **`erase()`** dùng để xóa phần tử mà con trỏ p đang trỏ tới. Cách thức xóa như sau:

- Nếu con trỏ p nằm ở nửa sau phần lưu trữ thì dời các phần tử sau p sang trái 1 vị trí.
- Nếu con trỏ p nằm ở nửa trước phần lưu trữ thì dời các phần tử trước p sang phải 1 vị trí.

Câu 11: Cài đặt phương thức **`sort()`** dùng để sắp xếp các phần tử theo tiêu chí linh động. Tiêu chí sắp xếp này sẽ được xác định khi gọi thực hiện phương thức.

2. ĐƠN THỨC VÀ ĐA THỨC

Câu 12: Khai báo lớp **Monomial** dùng để mô tả 1 đơn thức có dạng $f(x) = ax^n$ với **a** là hệ số và **n** là số mũ ($n \geq 0$).

Câu 13: Cài đặt các phương thức khởi tạo cho lớp **Monomial** gồm:

- Phương thức khởi tạo mặc định

- Phương thức khởi tạo một đơn thức từ hệ số và số mũ
- Phương thức khởi tạo sao chép

Câu 14: Cài đặt các toán tử cho lớp **Monomial** gồm:

- Toán tử gán (**`operator =`**)
- Toán tử nhân 2 đơn thức (**`operator *`**)
- Toán tử chia 2 đơn thức (**`operator /`**)

Câu 15: Cài đặt các phương thức

- Phương thức **`derivative()`** tính đạo hàm của đơn thức
- Phương thức **`inf_integral()`** tính nguyên hàm của đơn thức. Hằng số tự do C của nguyên hàm coi như bằng 0.

Câu 16: Sử dụng **CustomArray** và **Monomial** để khai báo lớp **Polynomial** dùng để mô tả 1 đa thức có dạng $f(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$. Có thể xem 1 đa thức là mảng của nhiều đơn thức

Câu 17: Cài đặt các phương thức khởi tạo cho lớp **Polynomial**:

- Phương thức khởi tạo mặc định
- Phương thức khởi tạo một đa thức từ mảng các đơn thức
- Phương thức khởi tạo một đa thức từ chuỗi ký tự biểu diễn đa thức (xem định nghĩa ở câu 17)
- Phương thức khởi tạo sao chép

Câu 18: Cài đặt các toán tử cho lớp **Polynomial**:

- Toán tử gán (**`operator =`**)
- Toán tử vị trí (**`operator []`**) trả về 1 đơn thức
- Toán tử cộng 2 đa thức, cộng 1 đa thức và 1 số thực (**`operator +`**)
- Toán tử trừ 2 đa thức, trừ 1 đa thức và 1 số thực (**`operator -`**)
- Toán tử nhân 2 đa thức, nhân 1 đa thức và 1 số thực (**`operator *`**)
- Toán tử chia lấy phần nguyên của 2 đa thức (**`operator /`**)
- Toán tử chia lấy phần dư của 2 đa thức (**`operator %`**)

Câu 19: Cài đặt các phương thức

- Phương thức **`derivative()`** tính đạo hàm của đa thức

- Phương thức **`inf_integral()`** tính nguyên hàm của đa thức. Hằng số tự do C của nguyên hàm coi như bằng 0.

Câu 20: Viết chương trình tính toán đa thức gồm các chức năng:

- Đọc danh sách các câu lệnh từ file input, mỗi câu lệnh là 1 chuỗi ký tự nằm trên 1 dòng. Cú pháp các câu lệnh như sau
- Câu lệnh một ngôi:

  `<Tên câu lệnh>(<đa thức>)`
- Tên và ý nghĩa câu lệnh
  - "**der**": tính đạo hàm đa thức
  - "**inf**": tính nguyên hàm đa thức
- Câu lệnh hai ngôi:

  `<Tên câu lệnh>(<đa thức 1>,<đa thức 2>)`
- Tên và ý nghĩa các câu lệnh:
  - "**add**": cộng 2 đa thức
  - "**sub**": trừ 2 đa thức
  - "**mul**": nhân 2 đa thức
  - "**div**": chia lấy phần nguyên 2 đa thức
  - "**mod**": chia lấy phần dư 2 đa thức
- Một đơn thức được biểu diễn bằng chuỗi ký tự có dạng

  *"[Dấu của đơn thức][Hệ số của đơn thức]**x**^[Số mũ thứ của đơn thức]"*
- Đa thức được biểu diễn bằng 1 chuỗi ký tự có dạng:

  "[Đơn thức 1][Khoảng trắng][Đơn thức 2][Khoảng trắng]…[Khoảng trắng][Đơn thức n]"
- Quy ước:
  - Đa thức luôn được viết theo thứ tự bậc của đơn thức giảm dần, tức là đơn thức bậc cao nhất được viết đầu tiên, bậc cao thứ 2 được viết tiếp theo,…
  - Các đơn thức có hệ số bằng 0 tự động được loại bỏ.

  Ví dụ:

  Đa thức $\mathbf{x^{100} - 3x^{50} + 2x - 1}$ sẽ được viết thành chuỗi :

  `"+1x^100 -3x^50 +2x^1 -1x^0"`

  Đa thức $\mathbf{-3x^6 - 6x^7 + 2x^9 - 10}$ sẽ được viết thành chuỗi:

  `"+2x^9 -6x^7 -3x^6 -10x^0"`

  Câu lệnh: **`"add(+1x^100 -3x^50 +2x^1 -1x^0,+2x^9 -6x^7 -3x^6 -10x^0)"`** có nghĩa là cộng đa thức $\mathbf{x^{100} - 3x^{50} + 2x - 1}$ và đa thức $\mathbf{-3x^6 - 6x^7 + 2x^9 - 10}$

- Sau khi đọc file input, tính toán kết quả theo các câu lệnh trong file rồi xuất các đa thức kết quả vào file output, mỗi dòng 1 đa thức.
- Lưu ý: các hệ số trong đa thức là số thập phân, sẽ được làm tròn lấy 01 chữ số sau dấu phẩy thập phân.
- Chương trình xử lý bằng tham số dòng lệnh có dạng như sau

  <File exe thực thi> <đường dẫn file input> <đường dẫn file output>

  Ví dụ: 1612001.exe D:\input.txt D:\output.txt

Ví dụ:

File input.txt

```
add(-2x^2 +3x -1x^0,+3x^3 +1x^2 +1x^0)

sub(-2x^2 +3x -1x^0,+3x^3 +1x^2 +1x^0)

mul(-2x^2 +3x -1x^0,+3x^3 +1x^2 +1x^0)

der(-4x^3 +13x -1x^0)

inf(-4x^3 +13x -1x^0)
```

File output.txt

```
+3x^3 -1x^2 +3x^1

-3x^3 -3x^2 +3x^1 -2x^0

-6x^5 +5x^4 -3x^2 +3x^1 -1x^0

-12x^2 +13x^0

-1x^4 +6.5x^2 -0.5x^1
```

3. BẢNG ĐIỂM

Câu	Điểm	Câu	Điểm	Câu	Điểm	Câu	Điểm
1	1	6	2	11	1	16	1
2	1	7	2	12	1	17	2
3	1.5	8	2	13	1	18	8.5
4	1.5	9	2	14	1	19	2
5	2	10	2	15	1	20	9.5

4. QUY ĐỊNH (CẦN ĐỌC KỸ)

- Bài làm nhóm, tối đa 3SV/nhóm. Thời gian làm bài lần 1: **3 tuần.**
- Các nhóm sẽ tiến hành vấn đáp tại phòng thực hành sau khi nộp bài lần 1 và có thêm 2 tuần để sửa chữa theo góp ý và vấn đáp lần 2. Điểm đồ án là điểm trung bình sau 2 lần vấn đáp.
- Nếu nhóm 3 thành viên, mỗi thành viên trong nhóm phải làm tối thiểu 15/45 điểm. Nếu nhóm 2 thành viên, mỗi thành viên trong nhóm phải làm tối thiểu 22/45 điểm. Nếu nhóm 1 thành viên, mỗi thành viên trong nhóm phải làm tối thiểu 45/45 điểm.
- **Không sao chép code của nhau. Tất cả bài làm giống nhau từ 70% trở lên đều bị 0 điểm, không quan tâm ai là tác giả. Các nhóm phải tự bảo vệ source code của mình, không cho mượn source code.**
- **Các SV bị xác định sao chép code sẽ bị gửi danh sách lên Khoa.**
- SV có thể tự thiết kế thêm các phương thức và toán tử nếu thấy cần thiết.
- Tạo project đặt tên là MSSV1_MSSV2_MSSV3 (sắp theo thứ tự tăng dần). Mỗi lớp phải có 1 cặp gồm file .h và .cpp. File .h dùng để khai báo lớp và file .cpp dùng để cài đặt
- Tại mỗi hàm cần có chú thích rõ: ý nghĩa của input, ý nghĩa của output, chức năng của hàm là gì
- Tại mỗi d vòng code phải có chú thích diễn giải dòng code này xử lý cái gì
- Cấu trúc bài nộp như sau: Tạo thư mục gốc tên MSSV1_MSSV2_MSSV3. Trong thư mục gốc tạo 3 thư mục con: Release, Document, Source
    - Thư mục Release: chứa file MSSV1_MSSV2_MSSV3.exe chạy chương trình
    - Thư mục Source: chứa toàn bộ mã nguồn của đồ án
    - Thư mục Document: chứa file báo cáo (định dạng doc, docx hay pdf). Nội dung file báo cáo gồm:
        - Họ tên, MSSV của từng thành viên trong nhóm
        - Những câu đã làm được, những câu chưa làm được, thành viên nào làm được những câu nào
        - Hướng dẫn sử dụng, hướng dẫn chạy chương trình (có hình minh họa)
        - Video clip hướng dẫn (nếu có)
        - File báo cáo trình bày rõ ràng, mạch lạc, sạch đẹp
- Nén thư mục gốc thành file MSSV1_MSSV2_MSSV3.zip hoặc MSSV1_MSSV2_MSSV3.rar rồi upload theo link nộp trên moodle.
- **Những bài làm không đúng quy định sẽ bị trừ điểm tùy theo mức độ**